Detekcija i praćenje pješaka u okviru autonomne vožnje

Jusup, Mario

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni diplomski studij Računarstvo

DETEKCIJA I PRAĆENJE PJEŠAKA U OKVIRU AUTONOMNE VOŽNJE

Diplomski rad

Mario Jusup

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju

Ocjena diplomskog rada na sveučilišnom diplomskom studiju

Ime i prezime pristupnika:	Mario Jusup
Studij, smjer:	Sveučilišni diplomski studij Računarstvo
Mat. br. pristupnika, god.	D1290R, 07.10.2022.
JMBAG:	0165081763
Mentor:	izv. prof. dr. sc. Ratko Grbić
Sumentor:	
Sumentor iz tvrtke:	Zvonimir Kaprocki
Predsjednik Povjerenstva:	doc. dr. sc. Denis Vranješ
Član Povjerenstva 1:	izv. prof. dr. sc. Ratko Grbić
Član Povjerenstva 2:	prof. dr. sc. Mario Vranješ
Naslov diplomskog rada:	Detekcija i praćenje pješaka u okviru autonomne vožnje
Znanstvena grana diplomskog rada:	**Umjetna inteligencija (zn. polje računarstvo)**
Zadatak diplomskog rada:	Detekcija pješaka i njihovo praćenje u video signalu koji se dobiva s kamere montirane na prednjoj strani vozila je vrlo važan zadatak u okviru autonomne vožnje. U okviru ovog diplomskog rada potrebno je napraviti pregled postojećih metoda za detekciju i praćenje pješaka. Izraditi vlastiti algoritam za detekciju i praćenje pješaka uz korištenje dostupnih sintetičkih i stvarnih označenih primjera te ga evaluirati na odgovarajući način (npr. CARLA simulator). Demonstrirati rad sustava detekciju i praćenje pješaka implementacijom na ugradbeni računalni sustav opremljen
Datum ocjene pismenog dijela diplomskog rada od strane mentora:	18.09.2024.
Ocjena pismenog dijela diplomskog rada od strane mentora:	Izvrstan (5)
Datum obrane diplomskog rada:	01.10.2024.
Ocjena usmenog dijela diplomskog rada (obrane):	Izvrstan (5)
Ukupna ocjena diplomskog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:	01.10.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 01.10.2024.

Ime i prezime Pristupnika:	Mario Jusup
Studij:	Sveučilišni diplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D1290R, 07.10.2022.
Turnitin podudaranje [%]:	7

Ovom izjavom izjavljujem da je rad pod nazivom**: Detekcija i praćenje pješaka u okviru autonomne vožnje**

izrađen pod vodstvom mentora izv. prof. dr. sc. Ratko Grbić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD 1

2. PROBLEM DETEKCIJE I PRAĆENJA PJEŠAKA 3

2.1. Problem detekcije objekata 3

2.1.1. Mjere koje se koriste za evaluaciju detektora objekata 4

2.1.2. Konvolucijske neuronske mreže 6

2.1.3. YOLO algoritam 7

2.1.4. Verzije YOLO algoritma 9

2.2. Problem praćenja objekata u videozapisima 10

2.2.1. SORT i DeepSORT algoritmi 12

2.3. Postojeća rješenja detekcije i praćenja pješaka 13

3. PREDLOŽENO RJEŠENJE ZA DETEKCIJU I PRAĆENJE PJEŠAKA U OKVIRU AUTONOMNE VOŽNJE 17

3.1. Korišteni alati i tehnologije 18

3.2. Izgradnja vlastitog skupa podataka na temelju postojećih skupova s označenim pješacima 18

3.3. Treniranje detektora pješaka zasnovanog na *YOLOv7* algoritmu 23

3.4. Algoritam za detekciju i praćenje pješaka 28

3.4.1. Algoritam za detekciju i praćenje pješaka temeljen na *SORT* algoritmu 28

3.4.2. Algoritam za detekciju i praćenje pješaka temeljen na *DeepSORT* algoritmu 30

3.5. Implementacija predloženog algoritma za detekciju i praćenje pješaka na ugradbenu platformu 32

4. EVALUACIJA PREDLOŽENOG RJEŠENJA ZA DETEKCIJU I PRAĆENJA PJEŠAKA U OKVIRU AUTONOMNE VOŽNJE 34

4.1. Opis skupa podataka korištenog za evaluaciju 34

4.2. Opis mjera korištenih za evaluaciju 37

4.3. Rezultati evaluacije predloženog algoritma za detekciju i praćenje pješaka 40

4.3.1. Evaluacija predloženog algoritma za detekciju i praćenje pješaka temeljena na *SORT* algoritmu 41

4.3.2. Evaluacija predloženog algoritma za detekciju i praćenje pješaka temeljena na *DeepSORT* algoritmu za praćenje 44

4.3.3. Evaluacija predloženog algoritma za detekciju i praćenje pješaka s obzirom na brzinu izvođenja 48

5. ZAKLJUČAK 51

LITERATURA 52

SAŽETAK 53

ABSTRACT 54

ŽIVOTOPIS 55

PRILOZI 56

1. UVOD

Automobilska industrija prolazi kroz revolucionarne promjene zahvaljujući razvoju umjetne inteligencije. Jedan od najvažnijih ciljeva ove industrije je razvoj potpuno autonomnih vozila koja će moći samostalno donositi odluke u složenim prometnim situacijama bez ljudske intervencije. Ključna stavka autonomnih vozila je sposobnost lociranja i predviđanja putanje kretanja drugih sudionika u prometu, posebno pješaka, koji predstavljaju dinamične i često nepredvidive sudionike u prometnom okruženju.

Moderna vozila opremljena su raznim senzorima, a jedan od osnovnih senzora je kamera. Detekcija predstavlja lociranje objekata od interesa na ulaznoj slici dobivenoj sa senzora iz čega se detektirani objekti mogu locirati u prostoru koji okružuje vozilo. Praćenje je predviđanje pozicije pojedinog detektiranog objekta u sljedećem okviru videozapisa. Detekcija i praćenje pješaka su od presudne važnosti za sigurnost autonomnih vozila. Pješaci su ranjivi sudionici u prometu, a njihovo pravovremeno uočavanje i praćenje omogućava vozilima da izbjegnu sudare i osiguraju sigurnu vožnju. Razvoj učinkovitih algoritama za detekciju i praćenje pješaka zahtijeva kombinaciju naprednih tehnika računalnog vida, strojnog učenja i obrada podataka u stvarnom vremenu.

Cilj ovog diplomskog rada je istražiti postojeće metode za detekciju i praćenje pješaka te razviti i evaluirati vlastiti algoritam za detekciju i praćenje pješaka. Razvijeni algoritam je nadalje implementiran na ugradbenu platformu *NVIDIA Jetson Nano* s ciljem evaluacije rada algoritma na računalno ograničenim platformama kakve se često koriste u okviru autonomne vožnje. Fokus je na analizi i primjeni suvremenih tehnika dubokog učenja u svrhu detekcije i praćenja pješaka u video signalima, poput *YOLO* (engl. *You Only Look Once*) [1] detektora, koji je poznat po svojoj brzini vršenja detekcije. Osim toga, istražit će se i moderne metode za praćenje objekata, kao što su *SORT* (engl. *Simple Online and Realtime Tracking*) [2] i *DeepSORT* [3], koje omogućuju praćenje pješaka kroz video sekvence. U svrhu razvoja algoritma za detekciju pješaka što boljih performansi, potrebno je *YOLO* detektor trenirati na vlastitom skupu podataka koji se sastoji od slika na kojima se nalaze pješaci i odgovarajućih oznaka pješaka.

Rad je strukturiran na sljedeći način. U drugom poglavlju dan je pregled postojećih metoda i algoritama za detekciju i praćenje pješaka. Posebna pažnja posvećena je naprednim metodama koje koriste duboke neuronske mreže. U trećem poglavlju napravljen je pregled postojećih skupova podataka koji se sastoje od digitalnih slika s označenim pješacima. Predstavljena je analiza i usporedba dostupnih skupova podataka. Opisana je i priprema vlastitog skupa podataka koji je nastao kao kombinacija nekoliko odabranih postojećih skupova. Nakon toga je opisan

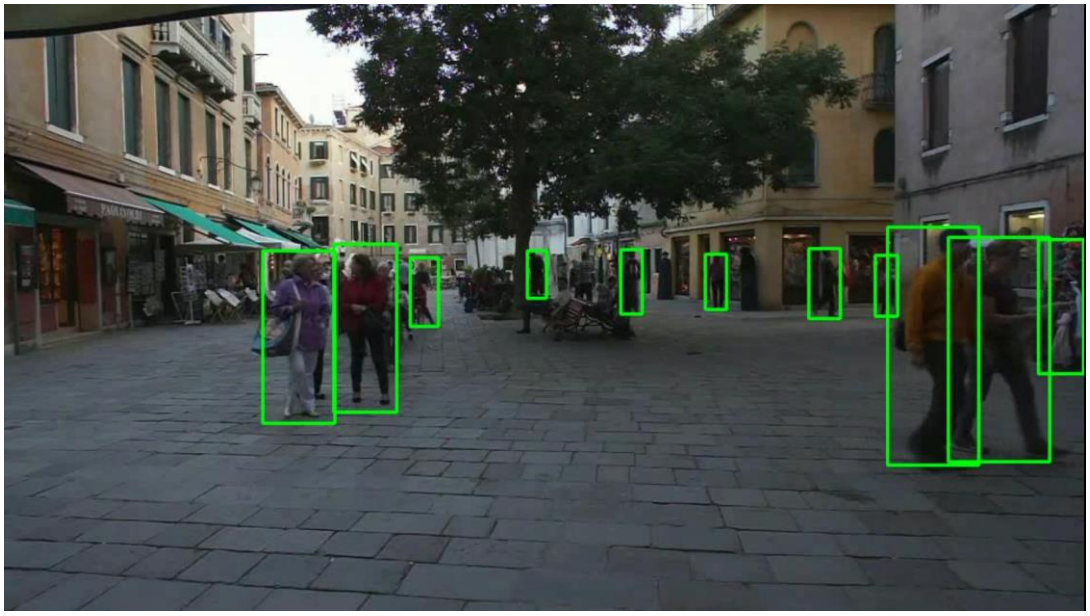
proces treniranja *YOLO* detektora objekata. Zatim je opisan algoritam koji kombinira detektor i algoritme za praćenje pješaka. Opisani su različiti načini rada algoritma i odabrani parametri. Na kraju poglavlja opisana je implementacija algoritma za detekciju i praćenje pješaka na ugradbeni računalni sustav. U četvrtom poglavlju predstavljena je evaluacija pojedinih algoritama na ugradbenom računalnom sustavu kao i na osobnom računalu. Peto poglavlje je zaključak rada.

2. PROBLEM DETEKCIJE I PRAĆENJA PJEŠAKA

Detekcija i praćenje objekata predstavljaju jedan od glavnih problema kod razvoja autonomnih vozila. U potpoglavlju 2.1. opisan je problem detekcije objekata. Opisane su konvolucijske neuronske mreže, *YOLO* algoritam i njegove verzije. U potpoglavlju 2.2. opisano je praćenje objekata, *DeepSORT* i *SORT* algoritmi. U potpoglavlju 2.3. predstavljena su neka od postojećih rješenja za detekciju i praćenje pješaka.

2.1. Problem detekcije objekata

Detekcija objekata u okviru autonomne vožnje predstavlja jedan od najvećih izazova jer zahtjeva najvišu razinu točnosti i preciznosti uz izvođenje u stvarnom vremenu, osobito kada se radi o detekciji pješaka. Pješaci su vrlo specifični kada ih promatramo kao objekte koje je potrebno detektirati. Posebnosti pješaka su izgled, brzina i pravci kretnja i razni položaji tijela što dodatno otežava ispravne detekcije. Detekcija objekata u slikama općenito podrazumijeva prepoznavanje i lociranje objekata na slici ili u okviru videozapisa. Moguće je detektirati više objekata na jednoj slici. Rezultat detekcije je brojčani zapis koji predstavlja granični okvir (engl. *bounding box*) za svaki detektirani objekt. Rezultat algoritma za detekciju prikazan je na slici 2.1. gdje su vidljivi detektirani pješaci koji su označeni graničnim okvirima. Granični okvir se zapisuje kao uređena četvorka (x, y, w, h) pri čemu su x i y koordinate središta graničnog okvira, a w i h su širina i visina graničnog okvira. Značajni rezultati u detekciji objekata postižu se primjenom naprednih metoda dubokog učenja, posebno konvolucijskih neuronskih mreža (engl. *Convolutional neural network, CNN*) objašnjenim u potpoglavlju 2.1.2.



Sl. 2.1. Primjer jedne slike iz skupa [27] s detektiranim pješacima označenim zelenim graničnim okvirima

2.1.1. Mjere koje se koriste za evaluaciju detektora objekata

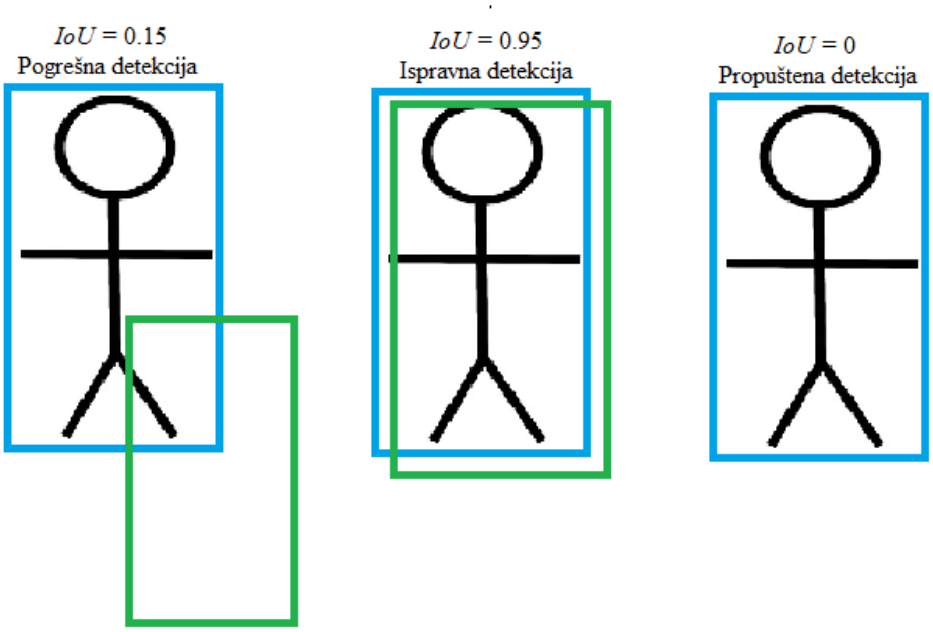
Kako bi se detektor objekata mogao evaluirati potrebne su slike koje imaju odgovarajuće oznake objekata od interesa, tipično u obliku graničnih okvira. Najčešće ovakve oznake nastaju pregledavanjem i ručnim označavanjem slika i smatraju se točnim pozicijama objekata na slici (engl. *ground truth*). Ispod su opisane neke od najznačajnijih mjera koje se koriste za evaluaciju detektora objekata: presjek preko unije (engl. *intersection over union, IoU*), preciznost, odziv, *F1* ocjena, srednja preciznost (engl. *average precision*, *AP*), *AP*@0.5, *AP*@0.5:0.95 i srednja prosječna preciznost (engl. *Mean Average Precision, mAP*).

- ***IoU***

  IoU je jedna od osnovnih mjera za evaluaciju performansi detektora objekata. Koristi se kako bi se odredilo koliko dobro detektor objekata prepoznaje objekte od interesa. Izraz za računanje mjere *IoU* dan je u (2-1). Računa se kao omjer površine presjeka graničnog okvira detekcije sa stvarnom oznakom objekta i površine unije ta dva granična okvira.

$$IoU = \frac{površina\ presjeka}{površina\ unije} \tag{2-1}$$

  IoU može imati vrijednosti između 0 i 1. Kada *IoU* ima vrijednost 1 to znači da se detekcija dobivena detektorom i stvarni objekt u potpunosti preklapaju, a kada ima vrijednost 0 to znači da nema preklapanja. Da bi se odredilo je li detekcija točna (engl. *True positive, TP*) ili pogrešna (engl. *False positive, FP*) potrebno je odrediti prag. Postoji i propuštena detekcija (engl. *False negative, FN*) kada stvarnom objektu na slici, odnosno njegovoj oznaci ne odgovara niti jedna detekcija. Uobičajeni prag iznosi 0.5 (50% preklapanja), ali može imati i druge vrijednosti, primjerice 0.75 (75% preklapanja), ovisno o specifičnostima problema. Na slici 2.2. je slikoviti prikaz određivanja *FP*, *TP* i *FN* uz korištenje *IoU* praga primjerice 0.5. Plavim graničnim okvirom prikazane su stvarne oznake objekta, a zelenim graničnim okvirom detekcije detektora.



Sl. 2.2. Ilustracija *IoU* s detekcijama određenima redom: *FP*, *TP* i *FN*

- **Preciznost, odziv i *F1* ocjena**

Izrazi za preciznost (2-2), odziv (2-3) i *F1* ocjena (2-4) su:

$$preciznost = \frac{TP}{TP + FP} \tag{2-2}$$

$$odziv = \frac{TP}{TP + FN} \tag{2-3}$$

$$F1 = 2 * \frac{preciznost * odziv}{preciznost + odziv} \tag{2-4}$$

gdje je:

- *TP* (engl. *true positive*) – broj ispravnih detekcija algoritma
- *FP* (engl. *false positive*) – broj objekata koje je algoritam detektirao, a u stvarnosti ne postoje
- *FN* (engl. *false negative*) – broj propuštenih detekcija.

- **Srednja preciznost, *AP*@0.5, *AP*@0.5:0.95**

AP se izračunava kao površina ispod krivulje kojoj je na *x* osi odziv, a na *y* osi pripadna preciznost, obje s vrijednostima od 0 do 1 (engl. *Precision-recall curve*). Krivulja predstavlja odnos između preciznosti i odziva pri različitim pragovima povjerenja (engl. *Confidence threshold*). Veće vrijednosti preciznosti obično su povezane s manjim vrijednostima odziva i obratno. *AP*@0.5 je prosječna preciznost izračunata s graničnom vrijednosti *IoU* od 0.5. *AP*@0.5:0.95 je prosjek *AP* mjeren na više pragova *IoU*, od 0.5 do 0.95 u koracima od 0.05. Ova mjera daje detaljniji uvid u performanse detektora objekata jer uzima u obzir više razina točnosti preklapanja.

- **Srednja prosječna preciznost**

Mjera *mAP* (engl. *Mean Average Precision*) se koristi za procjenu performansi detektora objekata. Ova mjera uzima u obzir preciznost i odziv u jednu mjeru kako bi se dobila cjelokupna slika učinkovitosti detektora objekata. Izračunava se izrazom (2-5) gdje je *n* broj vrsta objekata u skupu podataka, *APi* je prosječna preciznost za pojedinu vrstu objekata.

$$mAP = \frac{1}{n}\sum_{i=1}^{n} AP_i \tag{2-5}$$

2.1.2. Konvolucijske neuronske mreže

Konvolucijske neuronske mreže predstavljaju posebnu vrstu neuronskih mreža namijenjenih za rad sa složenim podacima, uključujući slike, tekst, zvuk i govor. Konvolucijske neuronske mreže koriste matematičku operaciju zvanu konvolucija [4], koja se primjenjuje na barem jednom sloju mreže. Konvolucija je specifična vrsta linearnog operatora koja omogućava filtriranje ulaznih podataka s ciljem izdvajanja ključnih značajki. Može se definirati za jednodimenzionalne i višedimenzionalne signale. U kontekstu obrade slika, dvodimenzionalna konvolucija je posebno važna. Na ulazni signal, kao što je slika, primjenjuje se filtar koji naglašava značajke poput rubova ili tekstura u izlaznom signalu. Inspirirane višeslojnim perceptronima, konvolucijske neuronske mreže su optimizirane za izvlačenje ključnih značajki iz složenih ulaznih podataka. Tipična konvolucijska neuronska mreža sastoji se od niza slojeva: ulazni sloj, konvolucijski sloj, sloj sažimanja i potpuno povezani slojevi. Ulazni sloj predstavlja ulaznu sliku i sadrži vrijednosti njenih piksela. Osnovna gradivna jedinica konvolucijskih neuronskih mreža je konvolucijski sloj i u njemu se pomoću raznih filtara obavljaju transformacije ulaznih podataka. Sloj sažimanja (engl. *pooling layer*) smanjuje dimenzije ulaznog podatka uz čuvanje većine važnih informacija. Potpuno povezani sloj (engl. *fully connected layer*) predstavlja posljednji sloj u mreži.

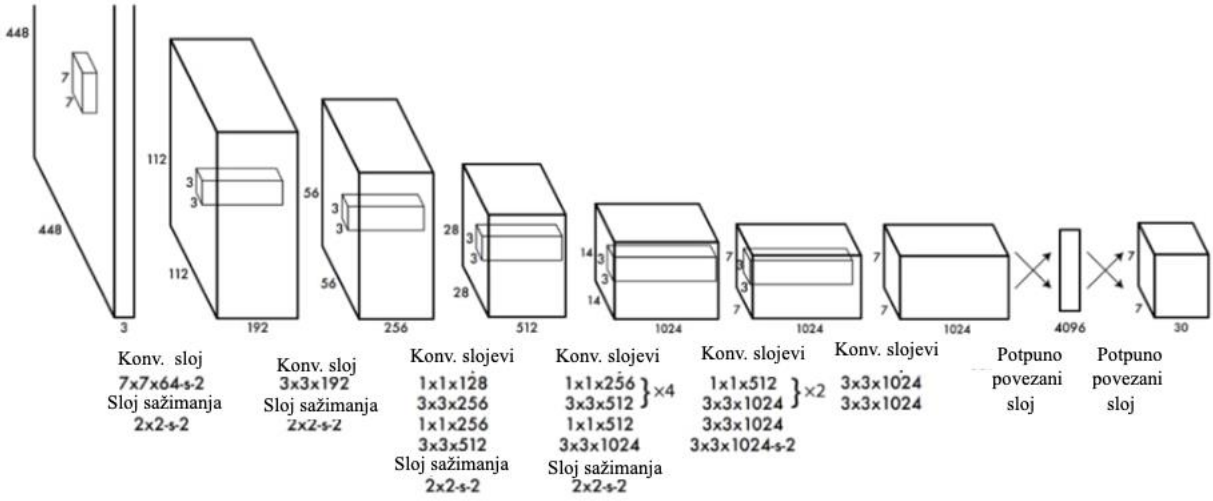
Jedna od najvažnijih komponenti konvolucijske neuronske mreže je aktivacijska funkcija. Aktivacijska funkcija je ključna komponenta da bi se uvela nelinearnost u model. Omogućuje mreži da uči složene oblike podataka i izvodi složene zadatke poput detekcije objekata.

Prilikom izgradnje konvolucijske neuronske mreže potrebno je odrediti njenu arhitekturu. Potrebno je odrediti broj, vrste i raspored slojeva, aktivacijskih funkcija i filtara. Nakon odabira arhitekture neuronske mreže potrebno je izvršiti trening na pripremljenom skupu podataka. Trening konvolucijskih neuronskih mreža je proces podešavanja težina tijekom kojeg mreža uči prepoznavati uzorke iz ulaznih podataka, kao što su slike, kako bi mogla točno detektirati ili prepoznavati objekte na temelju tih uzoraka. Za potrebe treninga potrebno je prikupiti odgovarajući skup podataka s pripadnim oznakama koje su u slučaju problema detekcije objekata granični okviri. Skup podataka potrebno je podijeliti na trening, validacijski i testni skup kako bi se model mogao pravilno trenirati i testirati njegova učinkovitost. Da bi mreža mogla učiti, potrebno je mjeriti koliko se predikcija modela razlikuje od stvarnih oznaka. Funkcija koja mjeri tu razliku naziva se funkcija gubitka. Trening je iterativni proces kojim se minimizira funkcija gubitka i izvodi se zadani broj epoha. Radi na sljedećem principu. Ulazne slike prolaze kroz mrežu i vrše se predikcije. Zatim se izračunava gubitak između predikcije mreže i stvarnih podataka. Zatim se odgovarajućim algoritmom optimizacije ažuriraju težine mreže, npr. gradijentnom metodom. Propagacija unazad (engl. *backpropagation*) je metoda za računanje gradijenta gubitaka u odnosu na parametre mreže. Algoritam optimizacije koristeći gradijente gubitka ažurira parametre mreže, čime se smanjuje gubitak. Validacijski skup podataka služi kako bi se mreža tijekom treninga povremeno evaluirala i kako bi se spriječilo da se mreža previše prilagodi trening podacima (engl. *overfitting*). U tom slučaju model bi često vršio loše predikcije na novim, neviđenim podacima. Postoje dva osnovna načina treniranja duboke neuronske mreže: klasična metoda treniranja od nule (engl. *Without pre-trained weights*) i prijenosno učenje (engl. *Transfer learning, TL*). Klasična metoda učenja podrazumijeva treniranje mreže od samog početka, koristeći slučajno inicijalizirane težine. S druge strane, kod metode prijenosnog učenja koristi se unaprijed trenirana mreža koja je već ranije naučila korisne značajke na velikim skupovima podataka. Mreža prilagođava naučene značajke novom zadatku dodatnim treningom. U odnosu na klasičnu metodu, metoda prijenosnog učenja zahtijeva manji skup podataka za trening, manje resursa i potencijalno robusnije performanse.

2.1.3. YOLO algoritam

YOLO (engl. *You Only Look Once*) je popularan algoritam za detekciju objekata predstavljen 2016. godine i od tada je objavljeno nekoliko verzija algoritma. Poznat je po svojoj točnosti i brzini

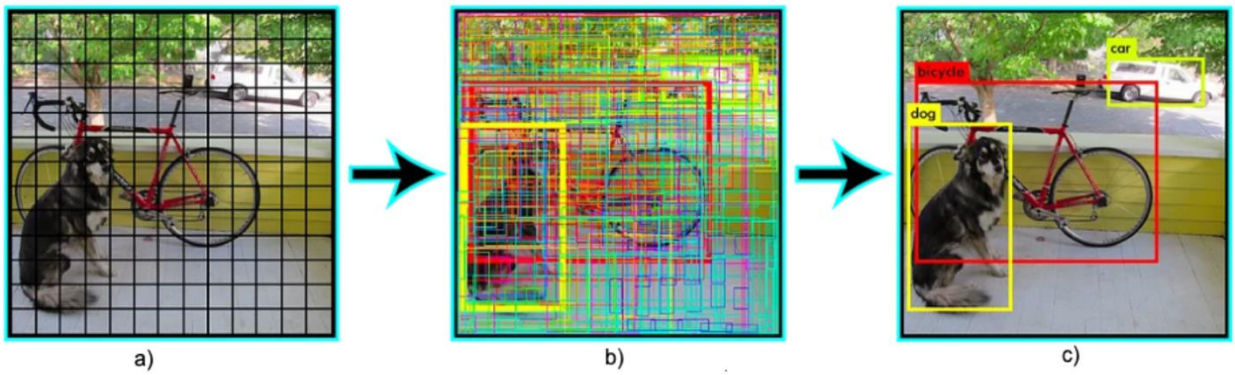
izvođenja. *YOLO* algoritam, za razliku od algoritama za detekciju temeljenih na prijedlozima regija (engl. *region proposal*), predstavlja pristup u kojem se u jednom prolasku ulazne slike kroz mrežu dobivaju detekcije objekata od interesa. Većina ostalih algoritama to čine u dva prolaska kroz mrežu zbog čega *YOLO* algoritam zahtjeva manje računalnih resursa i vremenski je učinkovitiji. *YOLO* algoritam je baziran na dubokim konvolucijskim neuronskim mrežama. Osnovna arhitektura *YOLO* mreže se sastoji od 24 konvolucijska sloja nakon kojih slijede 2 potpuno povezana sloja kao što je prikazano na slici 2.3.



Sl. 2.3. Osnovna YOLO arhitektura [1]

Na slici 2.4. je primjer detekcije objekata u slučaju kada su definirane tri klase od interesa: pas, bicikl i automobil. Algoritam najprije ulaznu sliku dijeli na *N*x*N* ćelija istih dimenzija kako je prikazano na slici 2.4. a). Algoritam zatim za svaku ćeliju predviđa jedan ili više objekata ako se središte njihovih graničnih okvira nalazi unutar te ćelije. Svaki granični okvir opisan je izrazom oblika (2-6) gdje *c* predstavlja klasu, primjerice 0 je pas, 1 bicikl i 2 automobil. Varijable *x*, *y*, *h* i *w* su redom koordinate središta i dimenzije graničnog okvira. Varijabla *p* predstavlja pouzdanost predikcije algoritma između 0 i 1. Nakon toga se primjenjuje tehnika potiskivanja ne-maksimalnih vrijednosti (engl. *non-maximum supression*). Ovom tehnikom se eliminira preklapanje više graničnih okvira za isti objekt. Zadržava se samo okvir s najvišom ocjenom pouzdanosti za pojedini detektirani objekt.

$$Y = [c, x, y, h, w, p] \tag{2-6}$$



Sl. 2.4. Ilustracija rada *YOLO* algoritma: (a) podjela slike na ćelije, (b) detekcije *YOLO* algoritma prije primjene tehnike potiskivanja ne-maksimalne vrijednosti, (c) izlaz *YOLO* algoritma

2.1.4. Verzije YOLO algoritma

YOLO algoritam je razvijen u više verzija gdje svaka iduća postiže bolje performanse i brzinu vršenja detekcija u odnosu na prethodnu. Kao što je ranije spomenuto, prva verzija *YOLOv1* [1] predstavljena je 2016. godine i predstavljala je prekretnicu u detekciji objekata jer vrši detekciju objekata u jednom prolazu što algoritam čini izuzetno brzim u odnosu na ostale algoritme za detekciju.

YOLOv2 [5] koji je poznat kao *YOLO9000*, druga je verzija *YOLO* algoritma. Novost u odnosu na prvu verziju je nova arhitektura mreže koja je nazvana *Darknet-19* koja se sastoji od devetnaest slojeva. Jedna od glavnih novosti je korištenje predviđenih graničnih okvira (engl. *anchor boxes*), to je skup predefiniranih graničnih okvira različitih dimenzija uz koje se postiže da algoritam detektira više objekata različitih veličina i oblika.

Značajna poboljšanja u odnosu na drugu verziju donosi *YOLOv3* [6] predstavljen 2018. godine. Treća verzija koristi novu arhitekturu mreže nazvanu Darknet-53 koja se sastoji od pedeset i tri konvolucijska sloja i optimizirana je za bolju detekciju objekata. Još jedna novost je uvođenje mreže piramide značajki (engl. *feature pyramid network, FPN*). Glavna ideja *FPN* tehnike je da se iz ulazne slike izgradi hijerarhijska piramida značajki gdje svaki nivo piramide predstavlja različitu razlučivost čime se omogućuje detekcija objekata različitih veličina. Korištenje *FPN*-a omogućuje mreži prepoznavanje malih objekata na slici koje ranije mreže nisu mogle prepoznati [7].

YOLOv4 [8] je predstavljen 2022. godine i donosi dodatna poboljšanja u donosu na prethodnu verziju. Baziran je na *CSPNet* (engl. *Cross Stage Partial Network*) arhitekturi mreže s pedeset i četiri sloja. Na različitim testovima YOLOv4 postiže iznimno dobre rezultate i postiže rad u stvarnom vremenu na konvencionalnim računalnim sustavima što ga čini prikladnim za široku uporabu.

YOLOv5 je razvijen od strane *Ultralytics* tima 2020. godine kao projekt otvorenog koda. Peta verzija YOLO algoritma koristi kompleksniju arhitekturu mreže nazvanu *EfficientDet*, zasnovanu na *EfficientNet* mreži čime se postiže veća točnost i bolja generalizacija na širi spektar kategorija objekata [9].

YOLOv6 [10] je predstavljen 2022. godine. Uvođenjem nove arhitekture, šesta verzija *YOLO* detektora postiže bolje performanse na testovima uz manju iskorištenost računalnih resursa.

YOLOv7 [11] predstavljen je također 2022. godine. Novost u arhitekturi je *E-ELAN* (engl. *extended efficient layer aggregation network*) što omogućuje bolje i učinkovitije učenje mreže. Temelji se na dizajnu koji optimizira putanju gradijenta. Putanja gradijenta se odnosi na način na koji se informacije o grešci, odnosno gradijenti prenose unatrag kroz mrežu tijekom procesa treniranja. Efikasna kontrola putanje gradijenta omogućava dubljoj mreži brže učenje, odnosno kraće vrijeme izvođenja treninga. Korištenjem tehnika proširenja, miješanja i spajanja kardinalnosti, *YOLOv7* može trenirati dublje mreže bez gubitka stabilnosti. Proširenje se odnosi na proširenje broja kanala i kardinalnosti, odnosno broja paralelnih grana unutar sloja mreže. Miješanje se odnosi na miješanje značajki u grupe nakon proširenja čime se postiže da različiti dijelovi mreže uče iz različitih značajki. Na kraju se značajke spajaju kako bi se zadržala izvorna struktura. Sedma verzija *YOLO* algoritma nadmašila je sve tadašnje algoritme za detekciju uz korištenje manje računalnih resursa i bolje performanse.

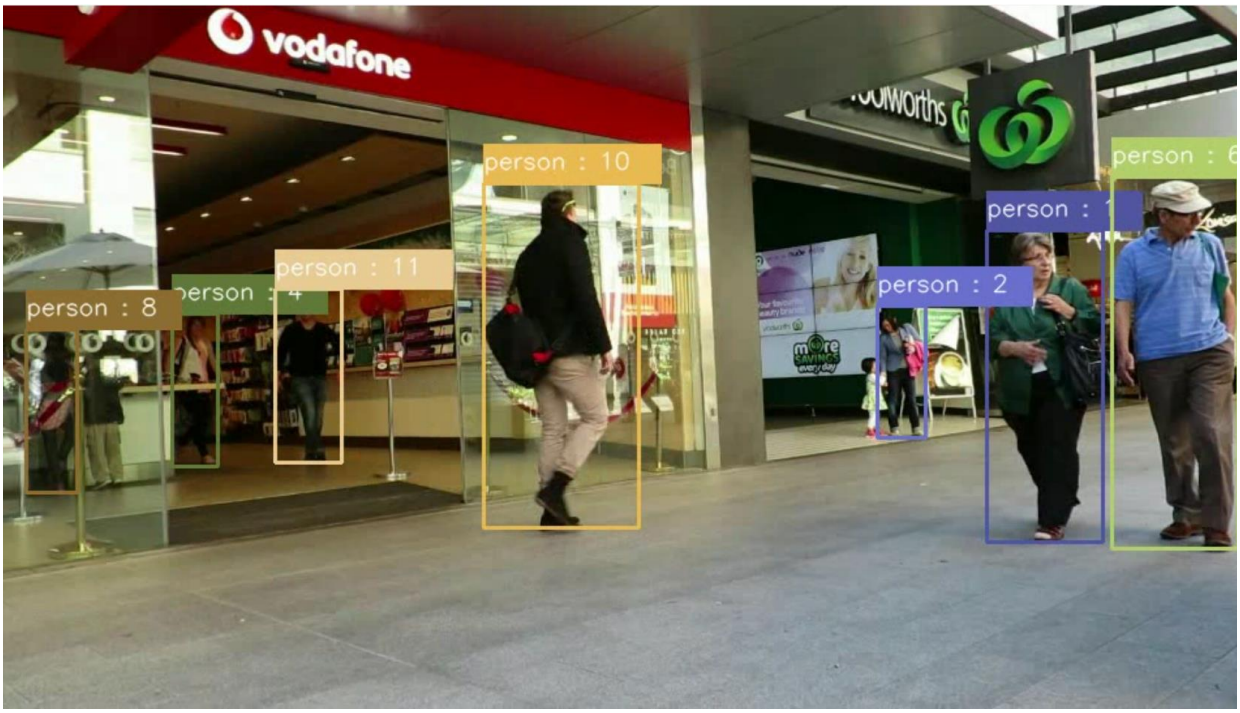
U vrijeme pisanja rada postoje još i novije verzije *YOLOv8, YOLOv9* i *YOLOv10*. Za potrebe rada korištena je verzija *YOLOv7* kao verzija s najvećom pozadinom dostupnih rješenja i podrškom zajednice.

2.2. Problem praćenja objekata u videozapisima

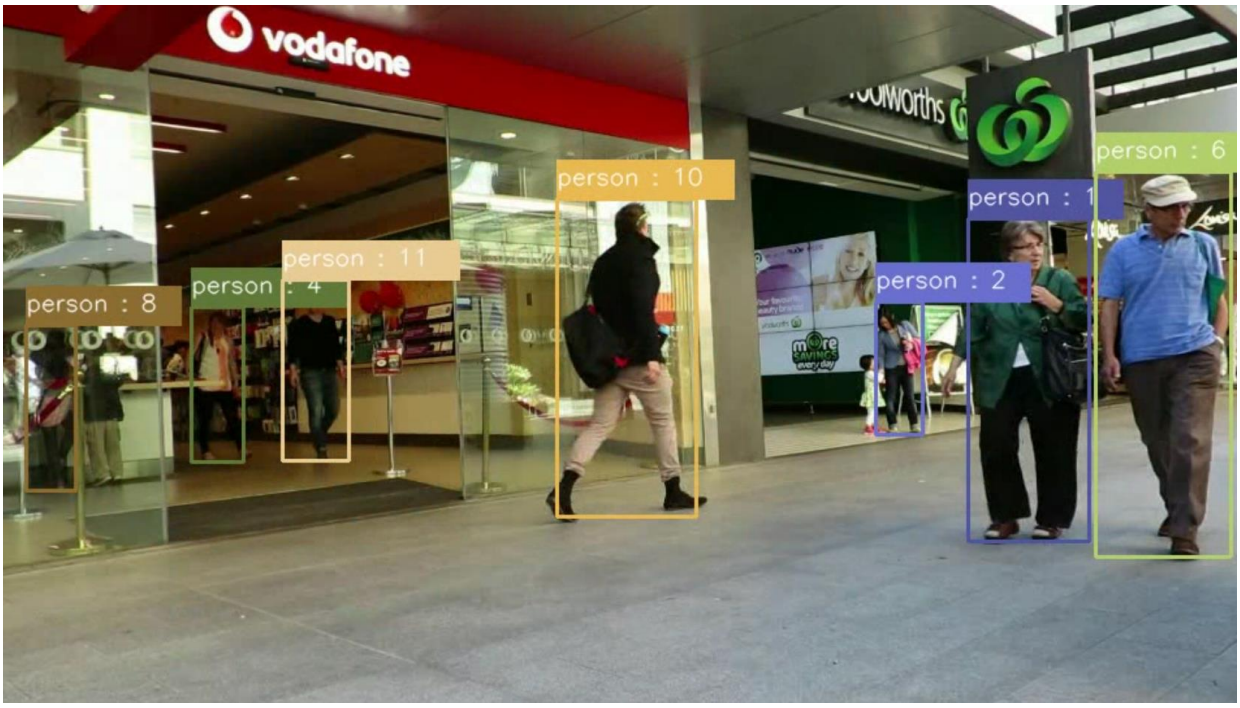
Praćenje objekata u videozapisima, osobito u kontekstu autonomne vožnje, predstavlja složen izazov koji uključuje kontinuirano prepoznavanje i praćenje pozicije objekata kroz niz uzastopnih okvira videozapisa. Nakon što je objekt detektiran algoritmom za detekciju, praćenje omogućuje sustavu da zadrži poveznicu između detektiranih objekata kroz vrijeme, čak i kada se njihova pozicija, veličina ili oblik mijenjaju. U kontekstu autonomne vožnje, praćenje objekata je od ključne važnosti jer omogućuje sustavu da predvidi njihovo kretanje i time osigura sigurnu interakciju između vozila i okoline. Problem praćenja objekata odnosi se na izazov prepoznavanja pozicije objekta u dinamičnom okruženju, što uključuje varijabilne uvjete kao što su promjene u osvjetljenju, djelomičnu okluziju i složene kretnje objekata. Osim toga, praćenje mora biti izuzetno učinkovito kako bi bilo izvedivo u stvarnom vremenu, što je presudno za primjene poput autonomne vožnje. Praćenje objekata obično se oslanja na detekciju u svakom okviru videozapisa,

ali i na algoritme za praćenje koji omogućuju povezivanje tih detekcija u različitim okvirima videozapisa. Na taj način sustav može identificirati koje detekcije iz različitih okvira pripadaju istom objektu. Glavni cilj praćenja je točno odrediti položaj i kretanje objekta kroz više okvira te minimizirati gubitke praćenja i lažne asocijacije između različitih objekata.

Postoje algoritmi specijalizirani za praćenje jednog objekta (engl. *Single object tracking, SOT*) i algoritmi za praćenje više objekata (engl. *Multiple object tracking, MOT*). Među najpoznatijim *MOT* algoritmima za praćenje su *SORT* (engl. *Simple Online and Realtime Tracking*) [2] i *DeepSORT* [3] objašnjeni u potpoglavlju 2.2.1. Slika 2.5. prikazuje princip rada praćenja objekata. Na slici 2.5. a) su graničnim okvirima i identifikacijskim oznakama označeni objekti od interesa koji se prate, a na slici 2.5. b) su označeni ti isti objekti nakon deset okvira videozapisa.



a)



b)

Sl. 2.5. Princip rada algoritma za praćenje, slika a) nastala je deset okvira videozapisa prije slike b)

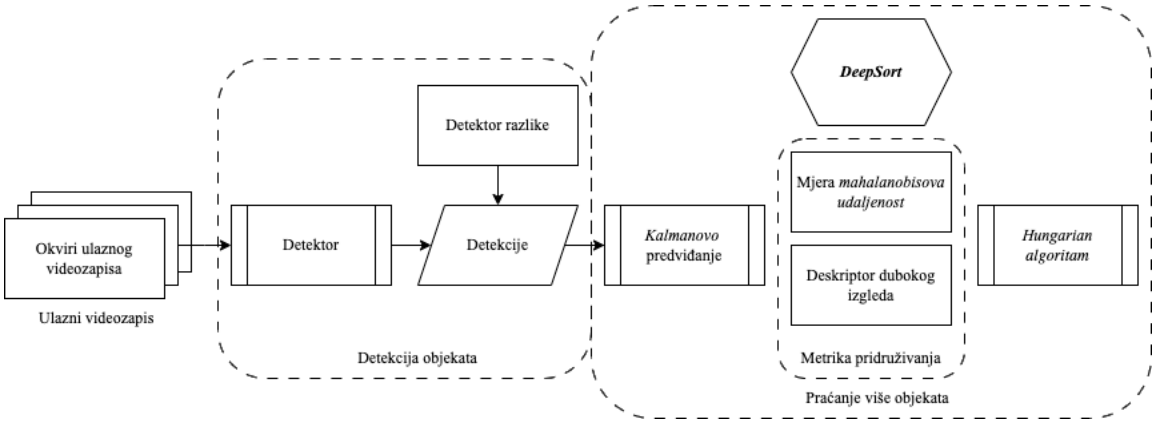
2.2.1. SORT i DeepSORT algoritmi

Za praćenje pješaka korišteni su algoritmi *SORT* i *DeepSORT* . Ovi algoritmi se rabe kod praćenja jednog ili više objekata u videozapisu.

SORT je algoritam za praćenje objekata koji na temelju detektiranih graničnih okvira iz prethodnih okvira videozapisa i brzine kretanja objekata predviđa poziciju objekta u idućem okviru videozapisa. *SORT* koristi *Kalmanove filtre* i *Hungarian algoritam*. *Kalmanov filtar* služi za predikciju kretanja objekta na temelju procjene konstantne brzine kretanja. *Hungarian algoritam* zatim povezuje objekte iz prethodnog okvira s objektima iz trenutnog okvira videozapisa. U slučaju da nema podudaranja objekata iz prethodnog i trenutnog okvira, objektu se dodjeljuje jedinstvena identifikacijska oznaka (*ID*). U slučaju podudaranja, objekt zadržava svoju dosadašnju oznaku. Također, ako u trenutnom okviru nedostaje pojedini objekt iz prethodnog tada se njegov *ID* uklanja.

DeepSORT je naprednija verzija *SORT* algoritma koja koristi duboke značajke za usporedbu objekata. *DeepSORT* uz metode iz *SORT* algoritma uključuje duboku neuronsku mrežu za ekstrakciju značajki iz isječaka objekata, što omogućuje precizniju usporedbu objekata kroz okvire, čak i u situacijama s velikom gustoćom objekata i složenim kretanjem [12]. Ilustracija rada ovog algoritma prikazana je na slici 2.6. Deskriptor dubokog izgleda (engl. *Deep appearance*

descriptor) izlučuje vizualne značajke iz svakog graničnog okvira. Te značajke su visokodimenzionalni vektori koji opisuju objekt unutar graničnog okvira. *Kalmanovo* predviđanje predviđa poziciju objekta unutar novog okvira videozapisa. *Hungarian algoritam* povezuje nove detekcije s postojećim praćenim objektima kombinacijom prostorne udaljenosti i udaljenosti značajki.



Sl. 2.6. Ilustracija rada *DeepSORT* algoritma

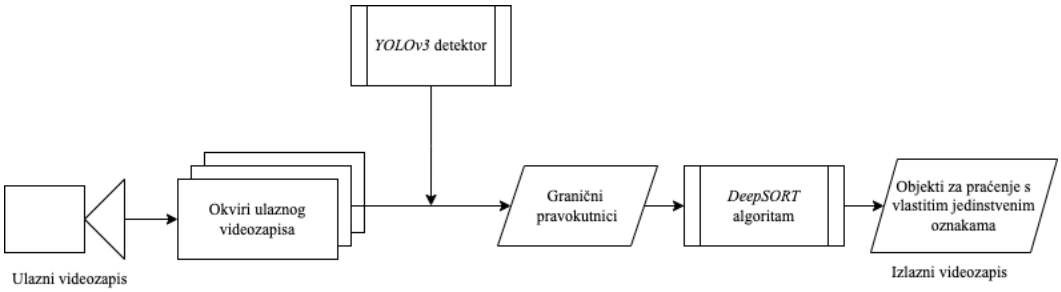
2.3. Postojeća rješenja detekcije i praćenja pješaka

Već duže vrijeme se u okviru razvoja autonomnih vozila razvijaju i rješenja za detekciju i praćenje pješaka. Postoji mnogo znanstvenih radova koji različitim pristupima i metodama rješavaju problem detekcije i praćenja pješaka. Važno je detaljno razumjeti različite pristupe kod razvoja ovakvih rješenja i prepoznati kako poboljšati učinkovitost rješenja. Postojeća rješenja za detekciju i praćenje pješaka koriste različite metode obrade slike, strojnog učenja i kombinacije senzorskih podataka, kao i različite načine implementacije na različite ugradbene računalne platforme. U nastavku je opisano nekoliko postojećih rješenja.

U radu [12] predstavljen je sustav za detekciju i praćenje pješaka temeljen na *YOLOv5* detektoru i *DeepSORT* algoritmu za praćenje. U rad je uvedena deformabilna konvolucijska mreža (engl. *Deformable Convolutional Network, DCN*) kako bi se poboljšala točnost određivanja graničnih okvira YOLOv5 detektora. Deformabilna konvolucijska jezgra omogućava konvolucijskim slojevima bolje prilagođavanje svojih receptivnih polja obliku objekta, u ovome slučaju pješaka, čime se postiže točnije lociranje. U ovom radu, deformabilna konvolucijska jezgra koristi se za izgradnju *Res-dcn* (engl. *Residual Deformable Convolutional Network*) komponenti koje zamjenjuju rezidualne komponente u dubokoj mreži *YOLOv5*. Rezidualne komponente, koje uključuju standardne konvolucijske slojeve, zamijenjene su deformabilnim konvolucijama kako bi se povećala fleksibilnost i preciznost detekcije, što je rezultiralo povećanjem prosječne točnosti detekcije za 3,39%, uz minimalan gubitak brzine obrade. Kako bi se poboljšala robusnost

algoritma u složenim pješačkim okruženjima, uvedena je fuzija *FHOG* (engl. *Fused Histogram Oriented Gradients*) i *CNN* značajki. *FHOG* je tehnika ekstrakcije značajki koja se koristi za prepoznavanje oblika i kontura objekata. Fuzija ovih značajki omogućava bolju prilagodbu na deformabilne objekte i složene situacije. Konkretno, *FHOG* značajke, koje su robusne na deformacije i promjene osvjetljenja, kombiniraju se sa značajkama dobivenim iz *CNN*-a kako bi se poboljšala ukupna točnost i robusnost algoritma za praćenje.

U radu [13] fokus je na detekciji i praćenju pješaka u urbanom okruženju radi poboljšanja autonomnih sustava poput autonomnih vozila. U radu je korišten *YOLOv3* detektor s *DeepSORT* algoritmom za praćenje. Slika 2.7. prikazuje princip rada ovog rješenja gdje se na svaki okvir videozapisa primjenjuje detekcija pomoću *YOLOv3* detektora. Dobivene detekcije se zatim predaju *DeepSORT* algoritmu na obradu. Objekti koje detektor ne prepoznaje zbog na primjer okluzije, i dalje se prate zahvaljujući algoritmu za praćenje.



Sl. 2.7. Princip rada rješenja predstavljenog u radu [13]

Rješenje se izvodi brzinom oko šesnaest okvira po sekundi (engl. *Frames per second, FPS*), testirano je na raznim videozapisima s pješacima i postiže visoke rezultate za preciznost 92% i odziv 97%.

U radu [14] kombiniran je *YOLOv3* s *ECO-HC* (engl. *Effective Convolution Operators Handcraft*) algoritmom za praćenje objekata u video signalima. Fokus u ovome radu je na praćenju pješaka s okluzijom, brzim kretanjima i promjenama položaja tijela što je inače vrlo izazovno za slične algoritme. *ECO-HC* predstavlja unaprijeđeni algoritam za praćenje pješaka koji koristi konvolucijske operatore kako bi poboljšao točnost i uspješnost praćenja u izazovnim uvjetima. Konvolucijski operatori su ručno optimizirani da budu efikasni u problemu praćenja pješaka i zadatak im je analiza vizualnih podataka. Algoritam je testiran na podskupu pješaka iz *OTB100* skupa podataka koji sadrži podatke s promjenama u osvjetljenju, okluzijama i raznim veličinama objekata. Algoritam se izvodi brzinom od dvadeset i četiri okvira po sekundi uz vrlo dobre performanse.

Rad [15] kombinira *YOLOv4* i *SORT* algoritam za praćenje pješaka. Algoritam *YOLOv4* je treniran na *VOC2007* skupu podataka gdje su ručno označeni granični okviri za automobile i

pješake. Detekcije *YOLOv4* algoritma predaju se *SORT* algoritmu koji zatim prati više objekata odjednom. U radu je napravljena usporedba rada algoritma s dvije različite verzije *YOLO* detektora, a to *su YOLOv3* i *YOLOv4*. Tablica 2.1. prikazuje usporedbu performansi *YOLOv4* i *YOLOv3* algoritama uz različite razlučivosti ulaznih slika. Mjerena je *mAP* mjera kao i broj okvira u sekundi. Vidljivo je kako *YOLOv4* postiže veću ocjenu *mAP* uz veći ili gotovo jednak broj okvira po sekundi.

Tablica 2.1. Rezultati usporedbe rada *YOLOv4* i *YOLOv3* algoritama [15]

Veličina ulazne slike	***YOLOv4***		***YOLOv3***	
	***mAP* (%)**	***FPS* (s)**	***mAP* (%)**	***FPS* (s)**
256x256	87.26	27.8	80.84	21.5
384x384	90.43	16.1	85.45	15.3
416x416	90.85	14.8	87.26	15.2
512x512	91.18	12.4	88.54	12.8

Iz tablice je vidljivo da promatrani algoritmi postižu vrlo dobre rezultate za *mAP* uz manji broj okvira po sekundi za slike veće razlučivosti i dobre rezultate za *mAP* i veći broj okvira po sekundi za slike manje razlučivosti.

U posljednjem promatranom rješenju [16] predstavljen je sustav za detekciju i praćenje pješaka temeljen na *Single Shot MultiBox Detector* (*SSD*) detektoru uz korištenje algoritma za praćenje. Dizajnirana je vlastita mreža temeljena na strukturi *SSD* detektora čiji su klasični slojevi sažimanja zamijenjeni slojevima spektralnog sažimanja. U tablici 2.2. prikazana je struktura modificirane mreže. Konvolucijski slojevi zapisani su kao *Conv*, a slojevi spektralnog sažimanja napisani su crvenom bojom.

Tablica 2.2. Prikaz strukture mreže u radu [16]

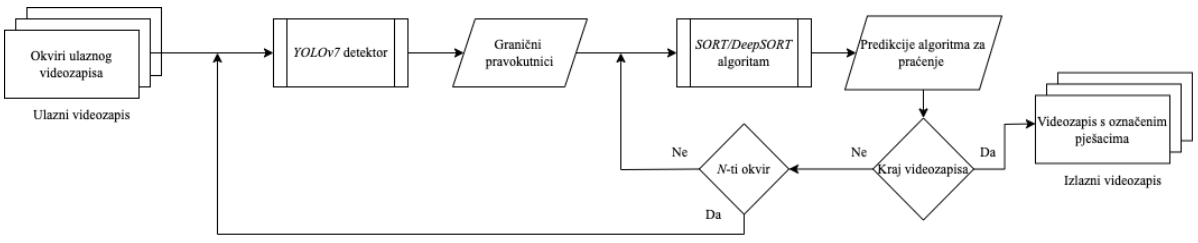
Tip	**Filtri**	**Korak**	**Izlazna vel.**	**Tip**	**Filtri**	**Korak**	**Izlazna vel.**
Conv1	32	3x3	300x300	**Conv5 2**	256	1x1	19x19
Spektralno sažimanje		2x2	150x150	**Conv5 3**	512	3x3	19x19
Conv2	64	3x3	150x150	**Conv5 4**	256	1x1	19x19
Spektralno sažimanje		2x2	75x75	**Conv5 5**	512	3x3	19x19
Conv3_1	128	3x3	75x75	**Spektralno sažimanje**		2x2	10x10
Conv3_2	64	1x1	75x75	**Conv6 1**	1024	3x3	10x10
Conv3_3	128	3x3	75x75	**Conv6 2**	512	1x1	10x10
Spektralno sažimanje		2x2	38x38	**Conv6 3**	512	3x3	5x5
Conv4_1	256	3x3	38x38	**Conv6 4**	256	1xl	5x5
Conv4_2	128	1x1	38x38	**Conv6 5**	256	3x3	3x3
Conv4_3	256	3x3	38x38	**Conv6 6**	128	1xl	3x3
Spektralno sažimanje		2x2	19x19	**Prosječno sažimanje**			3x3
Conv5_1	512	3x3	19x19	**Detekcija**			

Detektor pješaka, koji koristi mrežu iz tablice 2.2., modificiran je za rad u kompleksnoj frekvencijskoj domeni, čime se povećava preciznost detekcije. Algoritam za praćenje je implementiran kako bi kombinirao detekcije iz SSD mreže s dodatnim informacijama o kretanju objekata. Ovaj pristup koristi *Kalmanov filtar* za predikciju položaja objekata u sljedećem okviru, što poboljšava efikasnost praćenja. Sustav je testiran na *MOT16* skupu podataka i pokazao poboljšanje performansi detekcije i praćenja u odnosu na ostala dostupna rješenja.

3. PREDLOŽENO RJEŠENJE ZA DETEKCIJU I PRAĆENJE PJEŠAKA U OKVIRU AUTONOMNE VOŽNJE

Cilj ovog rada je kreirati i implementirati rješenje za detekciju i praćenje pješaka u okviru autonomne vožnje, koristeći napredne algoritme dubokog učenja i tehnike računalnog vida. Algoritam za detekciju pješaka zasnovan je na *YOLOv7* detektoru, koji je treniran i evaluiran na vlastitoj bazi podataka. Za praćenje pješaka u videozapisima korištena su dva različita algoritma: *SORT* i *DeepSORT*. U okviru rada potrebno je predloženi algoritam za detekciju i praćenje pješaka implementirati na ugradbenu platformu *NVIDIA Jetson Nano*. Implementacijom na ugradbenu platformu se simulira eventualna komercijalna upotreba zbog ograničenih resursa, a ujedno platforma pruža dovoljnu računalnu snagu za izvođenje složenih operacija detekcije i praćenja, čime se omogućuje primjena razvijenog rješenja u stvarnim uvjetima autonomne vožnje.

Slika 3.1. prikazuje način rada predloženog algoritma za detekciju i praćenje pješaka. Najprije se dohvaća ulazni okvir iz videozapisa te se provodi detekcija objekata na ovom okviru pomoću *YOLOv7* detektora. Zatim se detekcije pješaka šalju *SORT* odnosno *DeepSORT* algoritmu na obradu. Algoritam za praćenje generira predikcije poziciju graničnog okvira koji odgovara pojedinom pješaku u idućem okviru videozapisa. Varijabla *N* predstavlja broj za koliko će se okvira videozapisa ponovno izvršiti detekcija *YOLOv7* algoritmom. Umjesto da se detekcija pješaka vrši na svakom okviru ulaznog videozapisa, predlaže se rješenje koje uključuje detekciju pješaka u svakom 2., 5. ili 10. okviru, što su tri odvojena slučaja koja će biti detaljno evaluirana. Kada se ne provodi detekcija objekata, predloženi algoritam vrši praćenje pješaka. Postupak se ponavlja do kraja videosignala. Ovaj pristup značajno smanjuje računalne i vremenske zahtjeve koji dolaze primjenom detekcije objekata. Dodatno, korištenjem algoritma za praćenje omogućava se efikasnije određivanje pozicije pješaka čak i kada su djelomično zaklonjeni, što dodatno povećava pouzdanost i robusnost sustava. Predloženo rješenje u zasebnu datoteku pohranjuje videozapis s iscrtanim graničnim okvirima prepoznatih pješaka i pripadajuću tekstualnu datoteku sa zapisima graničnih okvira.



Sl. 3.1. Dijagram načina rada predloženog algoritma za detekciju i praćenje pješaka

U potpoglavlju 3.1. detaljno su opisani korišteni alati i tehnologije, zatim je u potpoglavlju 3.2. dan opis korištenog skupa podataka. U potpoglavlju 3.3. opisano je treniranje *YOLOv7* detektora, nakon čega je u potpoglavlju 3.4. detaljno opisan predloženi algoritam koji kombinira *YOLOv7* sa *SORT* i *DeepSORT* algoritmima za praćenje. U potpoglavlju 3.5. ukratko je opisan postupak implementacije razvijenog algoritma na ugradbenu platformu.

3.1. Korišteni alati i tehnologije

Algoritam za detekciju i praćenje pješaka razvijen je u programskom jeziku *Python* na osobnom računalu s operacijskim sustavom *Windows 10*. Python je pogodan jezik za razvoj ovakvog rješenja zbog svoje bogate kolekcije biblioteka za strojno učenje i računalni vid. Najvažnije korištene biblioteke uključuju:

- *OpenCV*: biblioteka za računalni vid koja omogućuje obradu i analizu video i slikovnih podataka.
- *TensorFlow*: platforma otvorenog koda za strojno učenje koja pruža podršku za duboko učenje i treniranje neuronskih mreža.
- *NumPy*: biblioteka za rad s nizovima i matematičkim operacijama, koja je temelj mnogih drugih biblioteka za računanje u Pythonu.
- *PyTorch*: biblioteka za duboko učenje koja omogućuje jednostavno treniranje modela dubokog učenja.

Implementacija algoritma za detekciju i praćenje pješaka izvršena je na ugradbenoj platformi baziranoj na *NVIDIA Jetson Nano* [17]. *NVIDIA Jetson N*ano je računalna platforma dizajnirana za primjenu u ugradbenim sustavima, pružajući podršku za napredne značajke umjetne inteligencije. Platforma je opremljena integriranom grafičkom karticom što joj omogućava izvođenje računalno zahtjevnijih algoritama u stvarnom vremenu. Prije korištenja platforme potrebno je instalirati operacijski sustav koji se naziva *NVIDIA JetPack.* Sustav je baziran na *Ubuntu 18.04.* verziji i uključuje sve potrebne alate i biblioteke za razvoj i implementaciju aplikacija baziranih na strojnom učenju.

3.2. Izgradnja vlastitog skupa podataka na temelju postojećih skupova s označenim pješacima

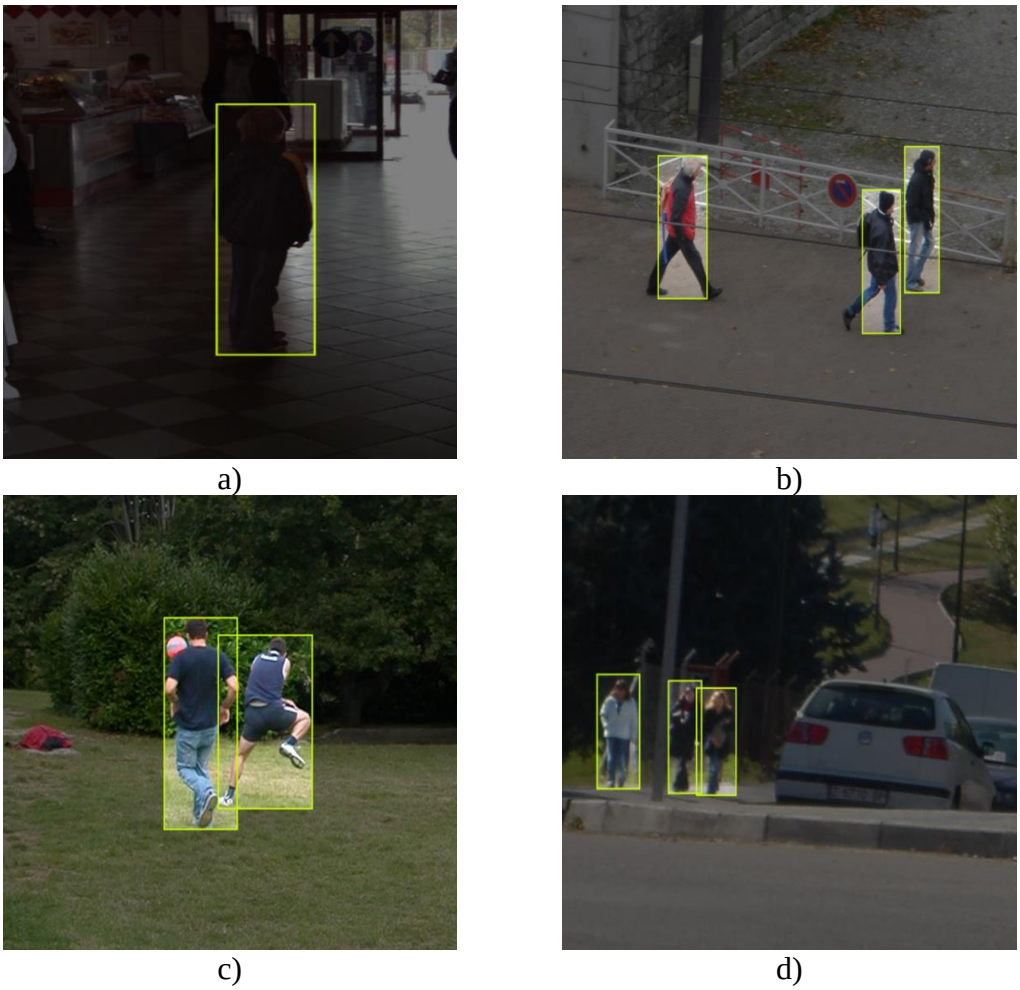
Za rješavanje problema detekcije pješaka primjenjuju se tehnike dubokog učenja. U ovome radu koristi se *YOLOv7* algoritam za detekciju kojeg je potrebno istrenirati na odgovarajućem skupu podataka. Za dobivanje robusnog *YOLOv7* detektora objekata potreban je pristup koji omogućuje modelu sposobnost prepoznati različite varijacije objekata u stvarnim uvjetima, čime se povećava njegova točnost i robusnost u stvarnom okruženju. U okviru ovog rada je stvoren skup

podataka sastavljen od nekoliko postojećih skupova kako bi se dobio velik i raznolik skup. Prvo je potrebno analizirati dostupne skupove podataka i utvrditi njihove prednosti i nedostatke kako bi se odabrali optimalni skupovi podataka za ovaj problem.

Prvi analizirani skup je *KITTI Vision Benchmark Suite* [18]. Skup je nastao 2012. godine i sastoji se od 15000 slika s ukupno 80000 označenih objekata. Razlučivost slika iz *KITTI* skupa podataka je 1280x384 piksela. Iz skupa je izdvojeno ukupno 1779 slika s 9400 pješaka označenih graničnim okvirima. Prednost ovog skupa je što sadrži označene pješake koji su zaklonjeni drugim objektima što može pridonijeti robusnosti modela. Na slici 3.2. su primjeri slika iz *KITTI* skupa. Pješaci su označeni plavim graničnim okvirima.

Sl. 3.2. Primjeri slika s označenim pješacima iz *KITTI* skupa podataka [18]

Drugi odabrani skup podataka je *INRIA* [19]. Ovaj skup se sastoji od 902 slike na kojima su graničnim okvirima označeni samo pješaci. Skup je odabran iz razloga što se sastoji od slika ljudi u vrlo raznolikim okruženjima, uslikanih iz različitih kutova i u različitim osvjetljenjima. Raznolikost *INRIA* skupa vidljiva je na slici 3.3. Uvjeti slabog osvjetljenja vidljivi su na slici a), različita perspektiva kamere vidljiva je na primjeru pod b), neobični položaji tijela ljudi iz primjera c) i ljudi koji su označeni u daljini na slici d).



Sl. 3.3. Primjeri slika s označenim pješacima iz *INRIA* skupa podataka [19]

Treći odabrani skup podataka je *EuroCity Persons* (*ECP*) skup [20]. Skup se sastoji od preko 47000 slika razlučivosti 1920x1024 piksela s preko 238000 oznaka pješaka u obliku graničnih okvira. Skup je sačinjen od slika iz 31 različitog europskog grada kroz sva četiri godišnja doba. Sadrži slike noću i danju, po suhim i mokrim vremenskim uvjetima što ga čini vrlo raznolikim i pogodnim za treniranje modela za ovaj rad. Iz razloga što je skup vrlo velik, odabrano je 4266 slika danju i 770 slika noću što je dovoljan broj za kreiranje vlastitog skupa. Slika 3.4. prikazuje četiri primjera slika iz ovog skupa s označenim pješacima zelenim graničnim okvirima. Na slici su vidljivi različiti vremenski uvjeti kao i uvjeti osvjetljenja.

Sl. 3.4. Primjeri slika s označenim pješacima iz *ECP* skupa podataka [20]

Kombinacijom ova tri skupa dobiven je vlastiti skup podataka koji se sastoji od ukupno 7717 slika. Vlastiti skup sadrži vrlo raznolike slike pješaka u raznim uvjetima što znači: dan i noć, sunčano i kišno vrijeme, veliki i mali pješaci u odnosu na veličinu slike i zaklonjeni pješaci. Tablica 3.1. prikazuje broj slika, broj označenih pješaka i razlučivost za pojedini odabrani skup podataka.

Tablica 3.1. Karakteristike odabranih skupova podataka

Skup podataka	**Broj slika**	**Broj označenih pješaka**	**Razlučivost (px)**
KITTI	1779	9400	1280x384
INRIA	902	2835	640x640
ECP	5036	27052	1920x1024

Za prilagodbu vlastitog skupa podataka potrebno je strukturirati slike i oznake u *YOLO* oblik. Svaka slika ima pripadnu datoteku istog naziva, ali s ekstenzijom .txt. Ova datoteka sadrži oznake u obliku graničnih okvira pri čemu je oznaka za svaki objekt na slici zapisana u novi redak datoteke. Izraz (3-1) je primjer jednog zapisa graničnog okvira jednog objekta.

<klasa objekta> <*x*> <*y*> <*širina*> <*visina*> (3-1)

Prvi broj predstavlja klasu kojoj pripada objekt u obliku cijelog broja. Za potrebe ovog rada koristi se jedna klasa (pješak) pa je to u ovome slučaju broj 0. Drugi i treći broj predstavljaju *x* i *y* koordinate središta graničnog okvira objekta. Četvrti i peti broj predstavljaju širinu i visinu graničnog okvira objekta. Važno je napomenuti da *x*, *y*, *širina* i *visina* trebaju imati decimalne vrijednosti u rasponu od 0 do 1. To se postiže skaliranjem dimenzija graničnog okvira s

dimenzijama slike. Primjer oznaka za sliku dimenzija 640x480 piksela dan je u tablici 3.2. Središte graničnog okvira za objekt klase 0 je u (x = 160, y = 225), a *širina* i *visina* iznose 80 odnosno 150 piksela. *Širina* i *x* koordinata središta se dijele sa širinom slike, a *visina* i *y* koordinata s visinom slike.

Tablica 3.2. Primjer oznaka za sliku s jednim pješakom prilagođenih u *YOLO* format

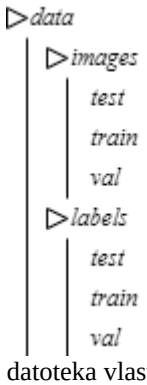
klasa objekta	***x***	***y***	***širina***	***visina***
0	0.25	0. 46875	0.125	0.3125

Nakon prilagođavanja oznaka objekata, odnosno anotacija, potrebno je vlastiti skup podijeliti u trening, validacijski i testni skup. Podjela je izvršena na način da trening skup sadrži 80%, validacijski i testni skup po 10% podataka. Od ukupno 7717, za trening skup je odvojeno 6171, za validacijski i testni skup po 773 slike. Podjela je izvršena na način da se od svakog zasebnog skupa (*KITTI, INRIA, ECP*) odvoji 80% slika za trening skup i po 10% slika za validacijski i testni skup pa raspodjela izgleda kao u tablici 3.3. Isto tako, od ukupnih 39000 oznaka pješaka, njih oko 80% je pripalo testnom skupu i po 10% je pripalo validacijskom i testnom skupu.

Tablica 3.3. Raspodjela slika u trening, validacijski i testni skup iz odabranih skupova podataka

Skup podataka	**Trening skup**	**Validacijski skup**	**Testni skup**
KITTI	1423	178	178
INRIA	720	91	91
ECP	4028	504	504
UKUPNO:	**6171**	**773**	**773**

Struktura datoteka vlastitog skupa podataka dana je na slici 3.5. U datoteci *images* nalaze se slike podijeljene u spomenuta tri skupa. U datoteci *labels* nalaze se oznake s istom strukturom kao i direktorij *images*. Svaka slika ima svoju odgovarajuću tekstualnu datoteku oznaka istog imena.



Sl. 3.5. Struktura datoteka vlastitog skupa podataka

3.3. Treniranje detektora pješaka zasnovanog na *YOLOv7* algoritmu

Nakon podjele skupa podataka na trening, validacijski i testni dio, može se krenuti na postupak treniranja *YOLOv7* detektora. Detektori su trenirani na 2 načina: klasičnom metodom bez pretreniranih težina i metodom prijenosnog učenja kako bi se dobiveni detektori mogli usporediti. *YOLOv7 tiny* je treniran metodom prijenosnog učenja jer se ta metoda pokazala uspješnijom pri treniranju originalnog detektora. Prvo je potrebno otvoriti alat *Naredbeni redak* na računalu i instalirati alat *pip* i virtualno okruženje koje služi za upravljanje paketima na računalu. Alat *pip* služi za instaliranje *Python* paketa, a on se instalira naredbom *sudo apt-get install python3-pip*. Zatim se instalira alat *Virtualenv* naredbom *sudo pip3 install virtualenv*. Virtualno okruženje naziva *yolov7* se kreira naredbom *virtualenv yolov7* i zatim aktivira naredbom *source yolov7/bin/activate*. Na ovaj način projekt se izolira od ostalih projekata na istom računalu. Zatim je potrebno pokrenuti naredbe sa slike 3.6.

Linija Kod

```
1:    git clone https://github.com/WongKinYiu/yolov7
2:    cd yolov7
3:    pip install -r requirements.txt
```

Sl. 3.6. Naredbe potrebne za pripremu *YOLOv7* za korištenje na lokalnom računalu

Prva linija klonira repozitorij *YOLOv7* na lokalno računalo. Drugom naredbom pozicioniramo se u direktorij sa svim *YOLOv7* datotekama. Posljednjom linijom instaliraju se sve potrebne biblioteke unutar kreiranog virtualnog okruženja.

Nakon prethodnog koraka potrebno je iz repozitorija [21] ručno preuzeti potrebne postojeće *YOLOv7* modele na lokalno računalo. Za izvođenje na osobnom računalu potrebno je preuzeti klasični unaprijed istrenirani model naziva *yolov7.pt*, a za izvođenje na ugradbenoj platformi potrebna je verzija s manje slojeva i parametara koja zahtjeva manje računalnih resursa naziva *yolov7-tiny.pt*. Zatim je potrebno napisati vlastitu konfiguracijsku datoteku sljedećeg sadržaja sa slike 3.7.

Linija Kod

```
1:    train: <putanja do trening skupa>
2:    test: <putanja do testnog skupa>
3:    valid: <putanja do validacijskog skupa>
4:    nc: 1 # broj klasa
5:    names: ["pedestrian"]
```

Sl. 3.7. Sadržaj konfiguracijske datoteke za treniranje detektora

Potrebno je postaviti putanje do vlastitog skupa podataka, a parametar *nc* predstavlja broj klasa koji za potrebe ovog rada iznosi jedan jer je potrebno detektirati samo klasu pješak. Nakon toga po potrebi treba podesiti hiperparametre za postupak treniranja mreže.

Za potrebe ovog rada mreža je trenirana na dva načina: klasičnim učenjem bez unaprijed istreniranog modela i metodom prijenosnog učenja kako bi se usporedile performanse pojedinog detektora. Trening se pokreće pokretanjem *python* skripte naredbom sa slike 3.8.

```
python train.py --workers 8 --device 0 --batch-size 8 --data
data/custom_data.yaml --img 640 640 --cfg cfg/training/yolov7-
custom.yaml --weights 'yolov7_training.pt' --name yolov7-transfer --
hyp data/hyp.scratch.custom.yaml –-epochs 100
```

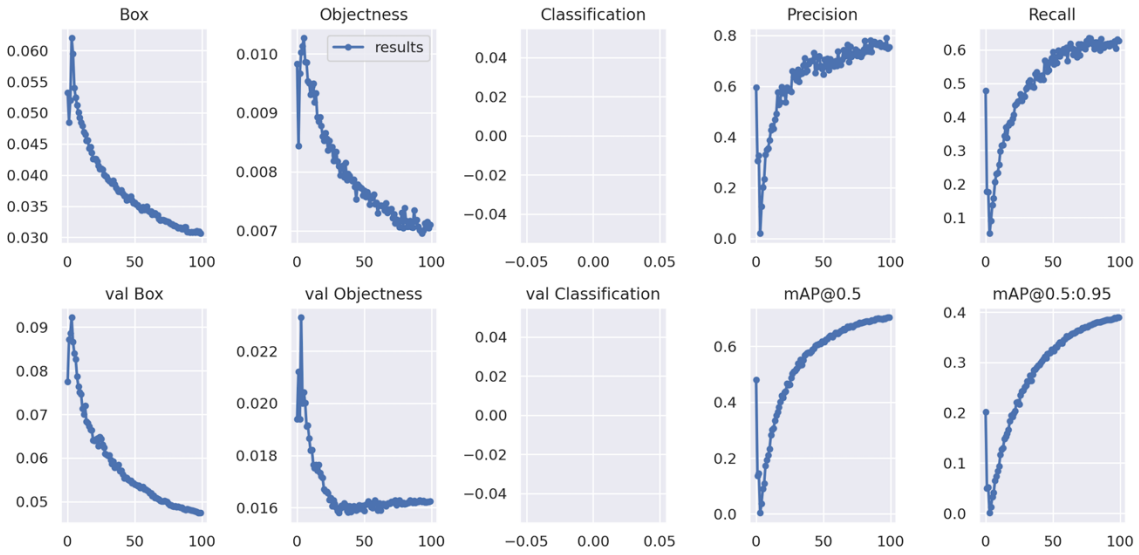
Sl. 3.8. Naredba za pokretanje treninga *YOLOv7* detektora

***workers* 8**: Ovaj parametar postavlja broj radnika (eng. *workers*) na 8. Radnici su niti ili procesi koji se koriste za učitavanje podataka tijekom treniranja, što može ubrzati proces treniranja.
***device* 0**: parametar specificira koji uređaj će se koristiti za treniranje. Vrijednost 0 označava da će se koristiti prvi grafička kartica. Ako se želi koristiti CPU, postavi se --device cpu.
***batch-size* 8**: parametar određuje se veličina skupine (eng. *batch size*) na 8. To je skup podataka koji se istovremeno prosljeđuju kroz mrežu tijekom podešavanja parametara mreže.
data data/custom_data.yaml: specificira putanju do ranije kreirane datoteke koja sadrži konfiguracijske podatke o skupu podataka koji se koristi za treniranje.
***img* 640 640**: parametar određuje dimenzije slika (širina i visina) koje se koriste prilikom treniranja mreže. U ovom slučaju, slike će biti skalirane na 640x640 piksela.
cfg cfg/training/yolov7-custom.yaml: parametar specificira putanju do YAML konfiguracijske datoteke koja sadrži arhitekturu modela koji će se trenirati.
weights 'yolov7.pt': ovdje se specificira putanja do prethodno treniranih težina (eng. *weights*) modela koje će se učitati prije početka treniranja. Ove težine mogu pomoći ubrzanju treniranja i postizanju boljih rezultata.
name yolov7-transfer: parametar daje ime trenutnom treningu, što je korisno za organizaciju i praćenje različitih rezultata treninga.
***hyp data/hyp.scratch.custom.yaml*:** parametar specificira putanju do ranije spomenute datoteke koja sadrži hiperparametre za treniranje, poput stope učenja, momentuma i drugih važnih postavki.
epochs 100: Ovim parametrom određuje broj potpunih prolazaka kroz cijeli skup podataka za treniranje.

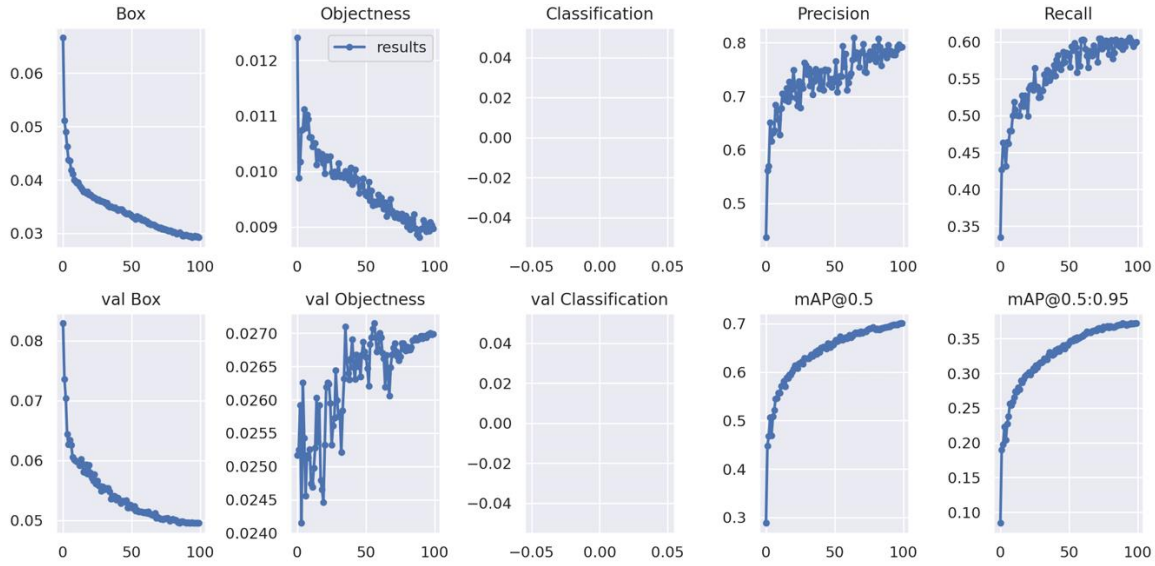
Za potrebe treninga *yolov7-tiny* modela metodom prijenosnog učenja potrebno je postaviti parametar –weights 'yolov7-tiny.pt' i -- cfg cfg/training/yolov7-tiny.yaml, a za potrebe treninga bez unaprijed istreniranog modela parametar --weights ' '. Treninzi se izvode na cijelom trening skupu podataka.

U procesu treninga su praćene metrike i gubici kako bi se ocijenile uspješnost treninga. Na slikama 3.9., 3.10., i 3.11. redom su prikazani rezultati za za *YOLOv7 TL, YOLOv7 tiny TL* i *YOLOv7 custom.*

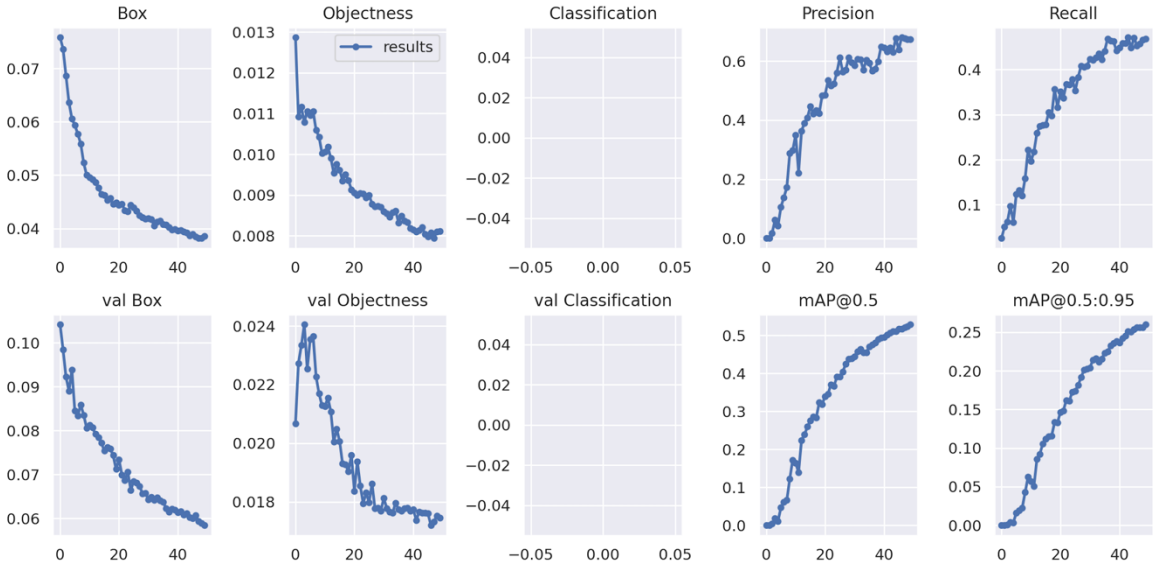
- ***Box*** mjeri koliko dobro model predviđa pozicije graničnih okvira u odnosu na stvarne pozicije tih objekata. Uključuje greške u predviđanju koordinata središta okvira (x, y) i dimenzija okvira (širina i visina). Cilj je minimizirati ovu vrijednost tijekom treninga kako bi predikcije bile što preciznije.
- ***Objectness*** mjeri koliko je model siguran da unutar predviđenog okvira postoji objekt. To je vjerojatnost da određeni okvir sadrži objekt (umjesto pozadine). Model uči razlikovati objekte od pozadine, a *objectness loss* kazni model ako predviđa da se objekt nalazi tamo gdje ga nema.
- ***Classification*** mjeri koliko dobro model klasificira objekt unutar okvira u ispravnu kategoriju (npr. automobil, pješak). Kazni model ako objekt unutar okvira bude pogrešno klasificiran ili ako bude pridružena pogrešna klasa. Cilj je smanjiti ovu vrijednost kako bi model točno prepoznavao klase objekata.
- ***val Box***: ovo je mjerenje *box* gubitka na skupu podataka za validaciju. Koristi se za procjenu kako model generalizira na nove, neviđene podatke. Ako je *val box* značajno veći od trening box gubitka, to može biti znak pretreniranja.
- ***val Objectness****:* mjeri *objectness* gubitak na validacijskom skupu. Pokazuje koliko je model siguran u postojanje objekata u validacijskom skupu, što je važno za generalizaciju.
- ***Val Classification****:* klasifikacijski gubitak na validacijskom skupu. Ako model dobro generalizira, *val classification loss* će biti sličan gubitku na trening skupu.
- *Precision, Recall, mAP@0.*5 i *mAP@0.5:0.95* su objašnjeni u drugom poglavlju.



Sl. 3.9. Rezultati metrika i gubitaka treninga za *YOLOv7 TL*



Sl. 3.10. Rezultati metrika i gubitaka treninga za *YOLOv7 tiny TL*



Sl. 3.11. Rezultati metrika i gubitaka treninga za *YOLOv7 custom*

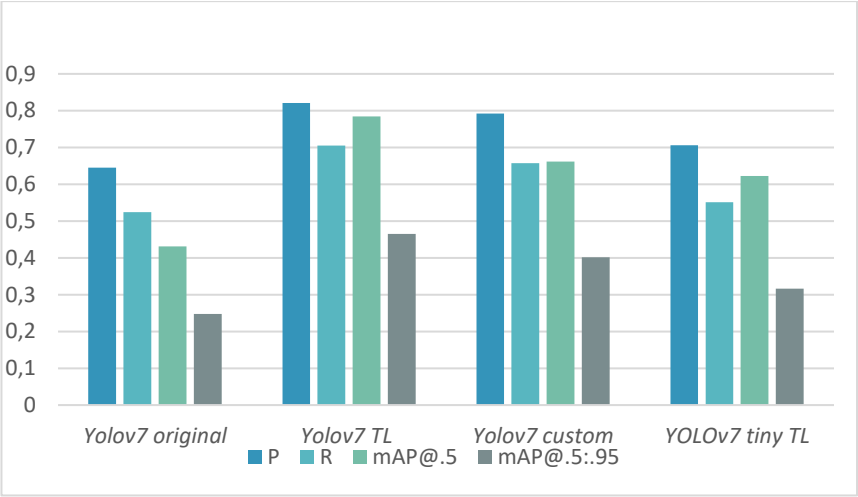
Iz grafova je vidljivo da su gubici i gubici na validacijskom skupu svedeni na minimum i više ne opadaju, stoga nije potrebno daljnje treniranje detektora da ne bi došlo do *overfittinga*.

Proces treniranja je unaprijed određen na 100 epoha za prijenosno učenje i 50 epoha za učenje trening bez unaprijed istreniranog modela. Brojevi epoha su određeni iz ranijih procesa treninga gdje su testirani hiperparametri treninga kako bi se spriječio *overfitting*. Prilikom treninga pohranjuju se težine za svaku epohu zasebno. Za predloženi algoritam korištene su težine s najboljim performansama temeljenim na evaluacijskim metrikama. Najbolje težine se automatski pohranjuju pod nazivom *best.pt* što označava najuspješniju iteraciju treninga. Nakon završenih treninga pojedinih detektora testirane su performanse detekcije na testnom skupu vlastitog skupa podataka. Originalni pretrenirani *YOLOv7* detektor nazvan je *YOLOv7 original*, detektor treniran metodom prijenosnog učenja nazvan je *YOLOv7 TL*, detektor treniran bez pretreniranih težina *YOLOv7 custom* i *YOLOv7 tiny* detektor izgrađen metodom prijenosnog učenja naziva se *YOLOv7 tiny TL*. Izmjerene su metrike predstavljene u drugom poglavlju: preciznost (*P*), odziv (*R)*, *mAP* pri *IoU* 0.5 i *mAP*@.5:.95. Rezultati testiranja prikazani su u tablici 3.4.

Tablica 3.4. Performanse pojedinih modela na testnom skupu podataka

Model	***P***	**R**	***mAP@.5***	***mAP@.5:.95***
YOLOv7 original	0,645	0,524	0,431	0,248
YOLOv7 TL	0,821	0,705	0,784	0,465
YOLOv7 custom	0,792	0,657	0,662	0,402
YOLOv7 tiny TL	0,706	0,551	0,623	0,316

Iz podataka tablice 3.4. je vidljivo kako detektor treniran metodom prijenosnog učenja postiže najbolje rezultate na testnom skupu podataka što je bilo i za očekivati. Na slici 3.12. grafički su prikazani podaci iz tablice. U odnosu na *YOLOv7* original, *YOLOv7 TL* postiže 17,6% bolju preciznost, 18,1% bolji odziv, 35,3% bolji mAP@.5 i 21,7% bolji mAP@.5:.95. *YOLOv7 tiny TL* koji je treniran metodom prijenosnog učenja unatoč manjem broju parametara i slojeva od originalnog *YOLOv7 original* detektora postiže bolje rezultate na testnom skupu.



Sl. 3.12. Usporedba odabranih metrika za evaluaciju detektora objekata na testnom skupu podataka za kreirane detektore

3.4. Algoritam za detekciju i praćenje pješaka

U ovom potpoglavlju predstavljeni su algoritmi za detekciju i praćenje pješaka, temeljeni na *SORT* i *DeepSORT* algoritmima za praćenje.

3.4.1. Algoritam za detekciju i praćenje pješaka temeljen na *SORT* algoritmu

Za pokretanje rješenja sa *SORT* algoritmom objašnjenog u dijelu 2.2.1. potrebno je najprije klonirati *SORT* repozitorij [22] u direktorij u kojem je ranije postavljen *YOLOv7*. Uvjeriti da je aktivno virtualno okruženje u kojemu su instalirane biblioteke potrebne za *YOLOv7* i zatim u naredbenom retku upisati naredbe sa slike 3.13.

Linija Kod

```
1:    git clone https://github.com/abewley/sort
2:    cd sort
3:    pip install -r requirements.txt
```

Sl. 3.13. Naredbe potrebne za početak korištenja *SORT* algoritma

Prvom linijom klonira se *SORT* repozitorij, zatim se promjeni radni direktorij u novo klonirani. Zadnjom linijom instaliraju se potrebne biblioteke za pokretanje *SORT* algoritma prikazane na slici 3.14.

Linija Kod

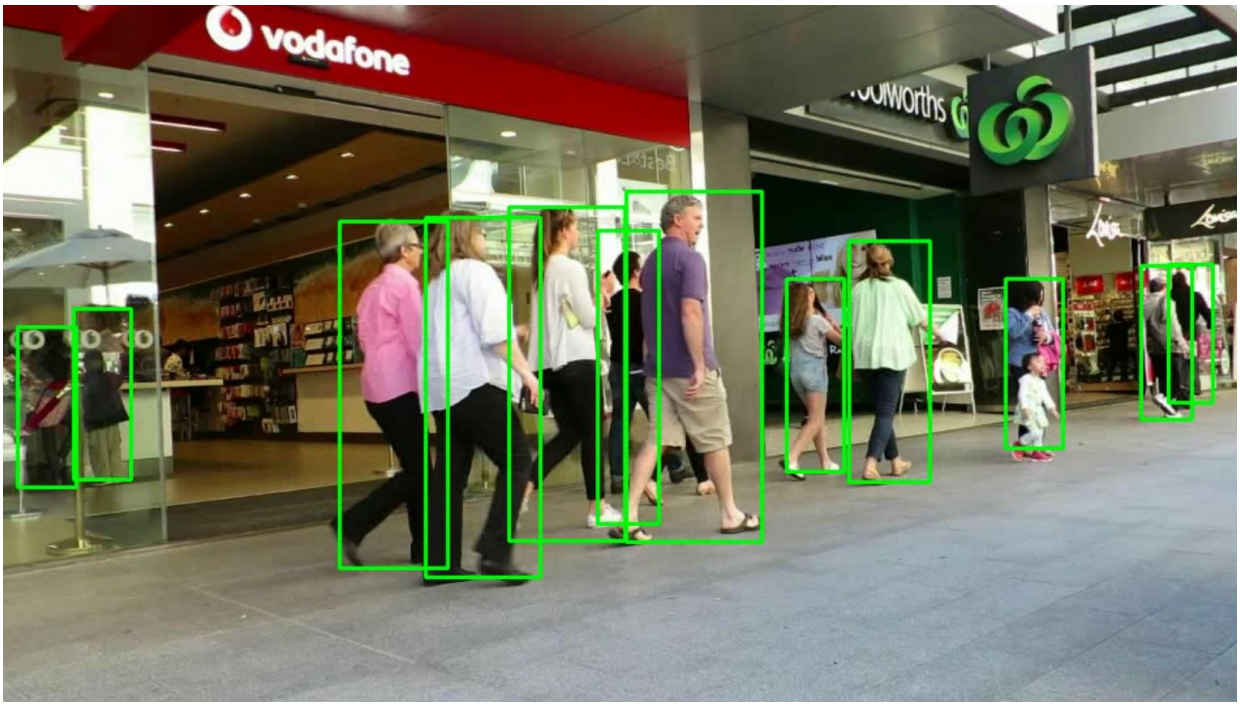
```
1:    filterpy==1.4.5
2:    scikit-image==0.17.2
3:    lap==0.4.0
```

Sl. 3.14. Biblioteke potrebne za *SORT* algoritam

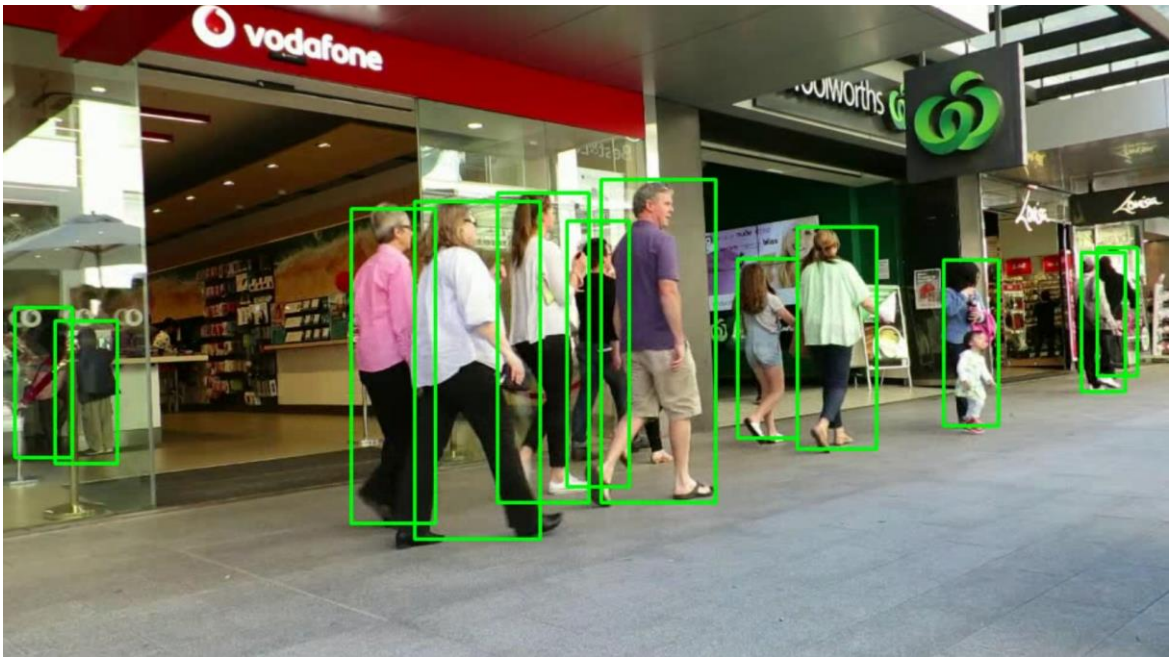
Algoritam je sada spreman za korištenje, a radi na principu da se kreira instanca *SORT* objekta koja obrađuje detektirane objekte u obliku graničnih okvira i zatim radi svoje predikcije. Za okvire videozapisa na kojima se ne vrši detekcija, algoritmu za praćenje se predaje vlastita predikcija u svrhu ažuriranja stanja sve dok se ne izvrši nova detekcija. Algoritam ujedno zapisuje rezultantni videozapis na kojemu su iscrtani granični okviri za svakog pješaka kao i tekstualnu datoteku sa zapisima detekcija u obliku graničnih okvira (slika 3.15.). Prva vrijednost predstavlja redni broj okvira videozapisa, zatim redom oznaka klase detektiranog objekta, *x* i *y* koordinate središta graničnog okvira, širina i visina, ostale vrijednosti su postavljene na 1 i -1 jer u ovome slučaju ne sadrže bitne informacije. Slike 3.16. a), b) i c) predstavljaju tri uzastopna okvira izlaznog videozapisa s iscrtanim graničnim okvirima za pješake koji se prate.

```
7,1,273.0,224.0,19.0,55.0,1,-1,-1,-1
7,1,537.0,243.0,26.0,54.0,1,-1,-1,-1
7,1,258.0,232.0,24.0,96.0,1,-1,-1,-1
8,1,292.0,222.0,46.0,133.0,1,-1,-1,-1
8,1,677.0,208.0,86.0,192.0,1,-1,-1,-1
8,1,220.0,221.0,61.0,141.0,1,-1,-1,-1
```

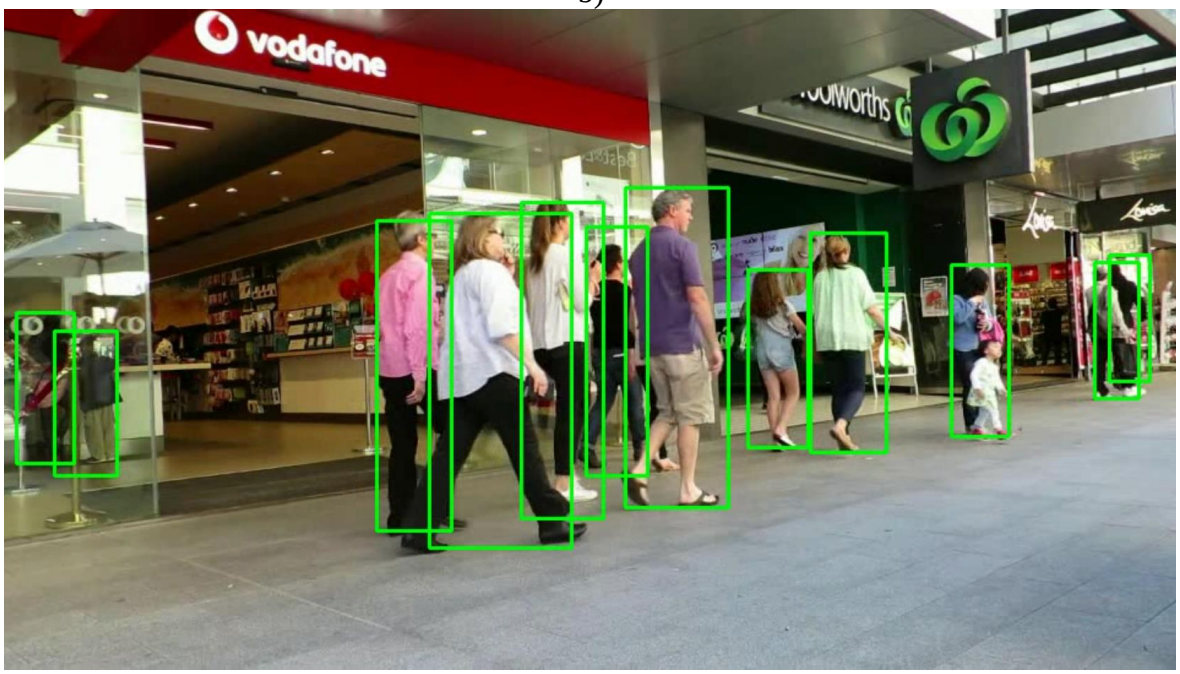
Sl. 3.15. Isječak tekstualne datoteke koja sadrži zapise graničnih okvira



a)



b)



c)

Sl. 3.16. Tri uzastopna okvira izlaznog videozapisa algoritma za detekciju i praćenje pješaka s *SORT* algoritmom

3.4.2. Algoritam za detekciju i praćenje pješaka temeljen na *DeepSORT* algoritmu

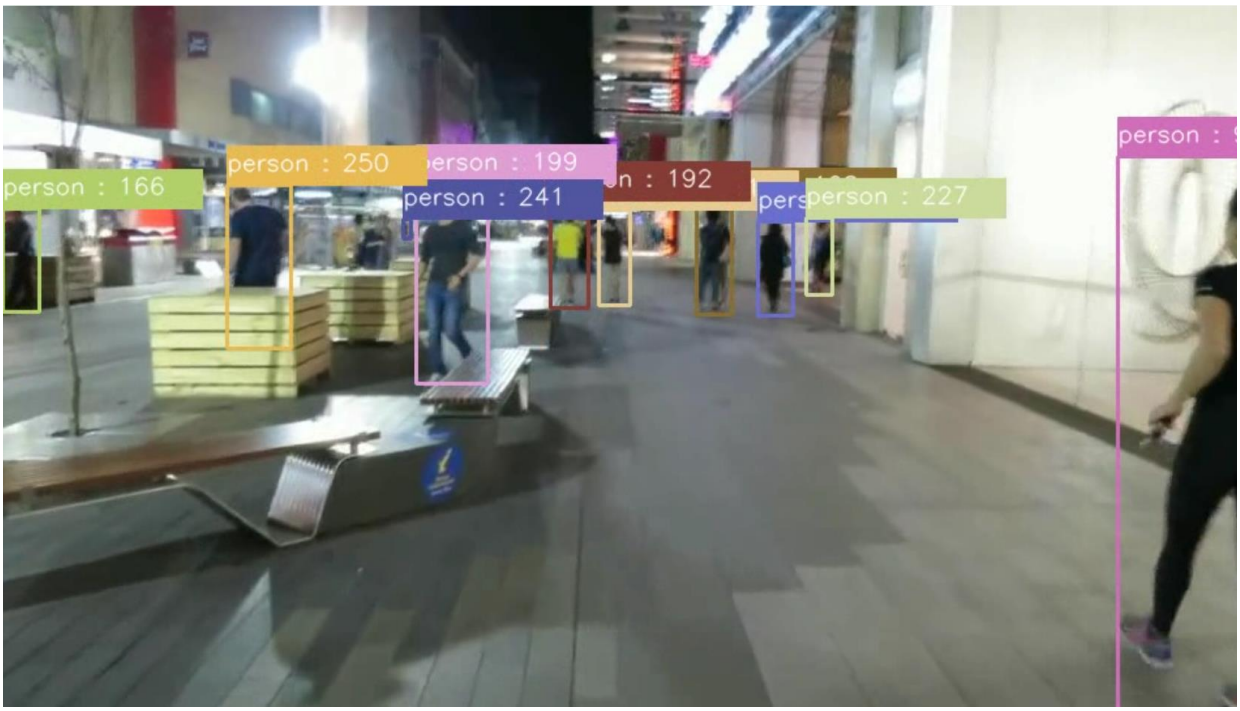
Za korištenje algoritma *DeepSORT* koji je objašnjen u potpoglavlju 2.2.1. potrebno je u direktorij u kojem je ranije kloniran *YOLOv7* klonirati i ovaj algoritam naredbom sa slike 3.17.

Linija Kod

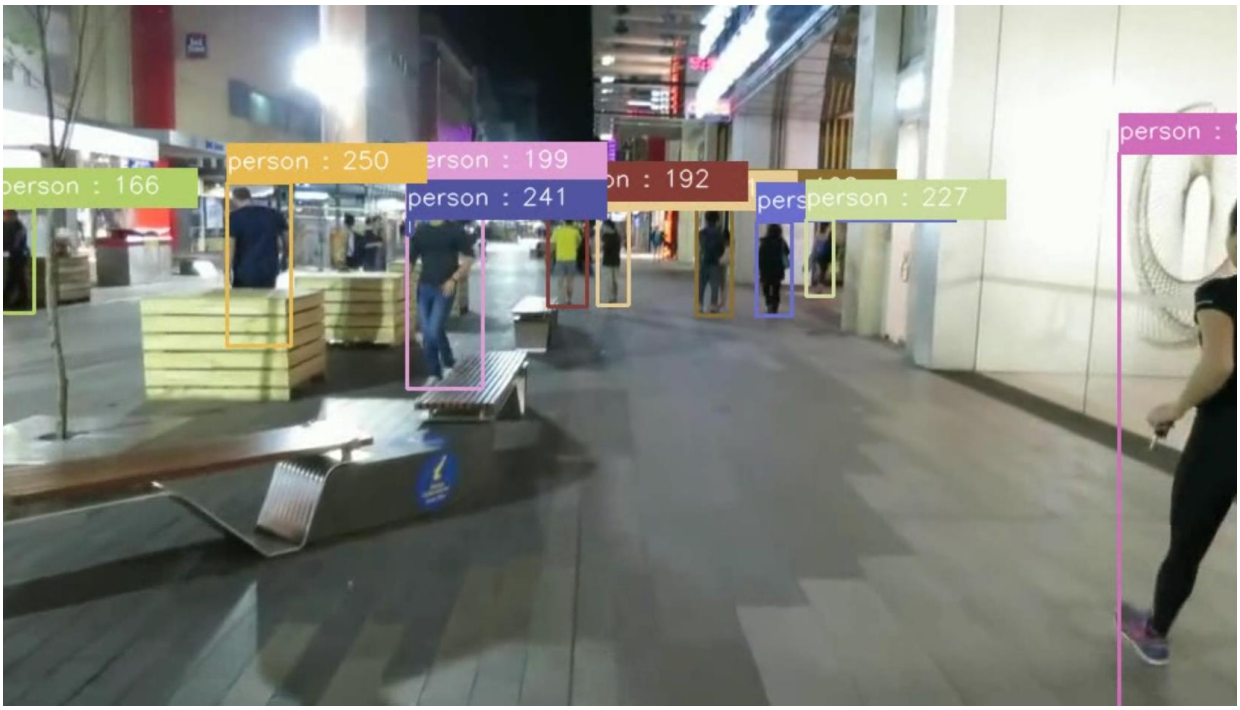
```
1:    git clone https://github.com/nwojke/deep_sort.git
```

Sl. 3.17. Naredba potrebna za korištenje *DeepSORT* algoritma

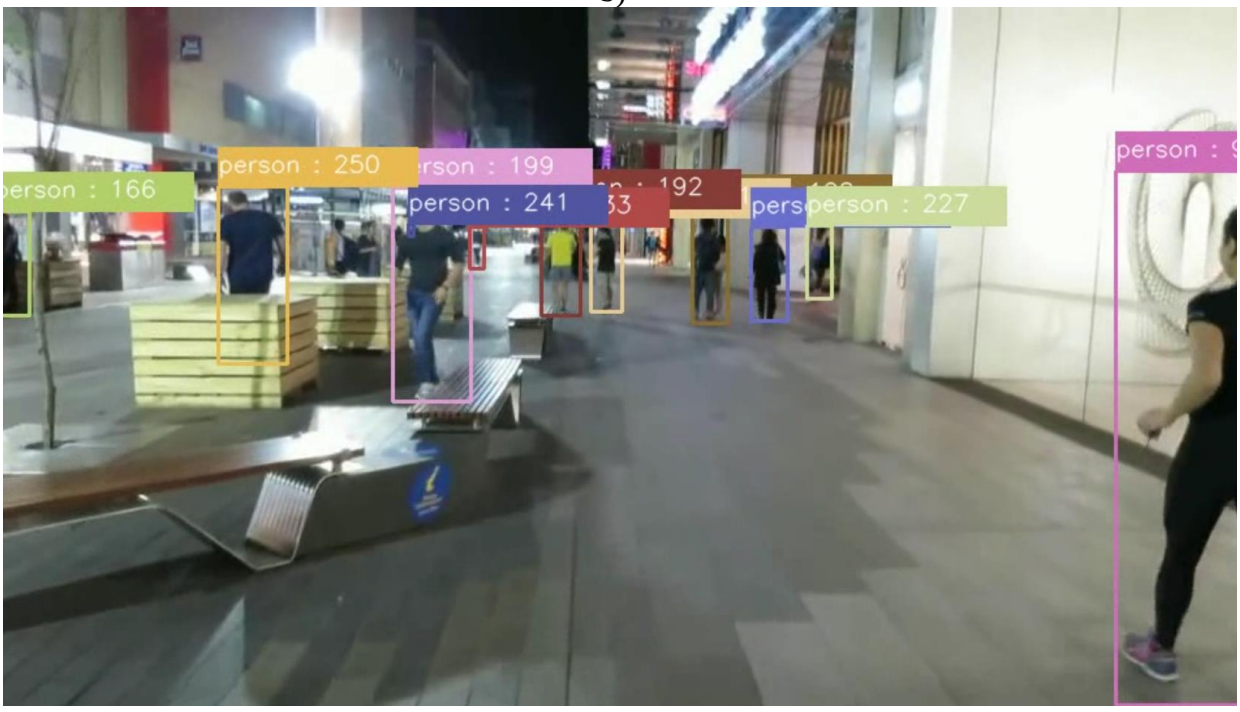
Potrebno je uvjeriti se da je aktivno virtualno okruženje s instaliranim *YOLO* bibliotekama pa zatim instalirati sljedeće biblioteke: *NumPy, sklearn, OpenCV, Tensorflow*. Ostatak algoritma radi po istom principu kao i *SORT* algoritam čiji je princip objašnjen u potpoglavlju 3.4.1. Jedina je razlika u načinu predavanja podataka za ažuriranje algoritmu za praćenje. Za razliku od *SORT* algoritma, *DeepSORT* algoritmu je potrebno predati i vektor značajke izgleda (engl. *Deep Appearance Descriptor*) jer ovaj algoritam osim na temelju brzine kretanja uspoređuje objekte i na temelju izgleda. Slika 3.18. pod a), b) i c) prikazuje 3 uzastopna okvira izlaznog videozapisa gdje su vidljivi granični okviri i jedinstvene oznake za svakog pješaka koji se prati.



a)



b)



c)

Sl. 3.18. Tri uzastopna okvira izlaznog videozapisa algoritma za detekciju i praćenje pješaka s *DeepSORT* algoritmom

3.5. Implementacija predloženog algoritma za detekciju i praćenje pješaka na ugradbenu platformu

Cilj je algoritme iz potpoglavlja 3.4.1. i 3.4.2. implementirati i na ugradbenu platformu. Na ovaj način simuliraju se stvarni uvjeti ograničenih resursa. Potrebno je na platformi s *Ubuntu*

18.04. operacijskim sustavom ponoviti sve korake kao na osobnom računalu s operacijskim sustavom *Ubuntu 22.04*. iz koraka 3.4.1. za *SORT* verziju i iz koraka 3.4.2. za *DeepSORT* verziju algoritma. U ovaj ugradbeni sustav spojena je *web* kamera pomoću koje je cilj testirati rad sustava. Za verziju sa *SORT* algoritmom potrebno je pokrenuti skriptu pisanjem *python3 detection_sort.py --source 1 --skip_frames 5* u naredbeni redak. Argument *source* predstavlja ulazni videozapis, u slučaju videozapisa s lokalnog računala ovdje je zapisana putanja do videozapisa, a u slučaju *web* kamere zapisuje se 1. Argument *skip_frames* predstavlja broj okvira koji će se preskočiti između dvije detekcije. Za pokretanje skripte s implementiranim *DeepSORT* algoritmom u naredbeni redak upisati *python3 detection_deepsort.py --source 1 --skip_frames 5*, dakle razlika je u nazivu skripte. Pokretanjem ovih algoritama vidljivo je da oba algoritma uspješno detektiraju i prate osobe u prostoriji koje su u kadru kamere. Detaljniji uvid u performanse rada pojedinog algoritma iznesen je u četvrtom poglavlju.

4. EVALUACIJA PREDLOŽENOG RJEŠENJA ZA DETEKCIJU I PRAĆENJA PJEŠAKA U OKVIRU AUTONOMNE VOŽNJE

U ovom poglavlju detaljno je opisan postupak evaluacije predloženog algoritma za detekciju i praćenje pješaka. Evaluacija je napravljena zasebno za rješenje koje koristi *SORT* i za ono koje koristi *DeepSORT* algoritam. U potpoglavlju 4.1. opisan je skup podataka koji je korišten za evaluaciju rješenja kao i priprema skupa za evaluaciju. U potpoglavlju 4.2. dodatno su opisane mjere korištene za evaluaciju. U potpoglavlju 4.3. izneseni su rezultati evaluacije predloženog algoritma za detekciju i praćenje pješaka zasebno za algoritam temeljen na *SORT* i *DeepSORT* algoritmima za praćenje i analizirana je brzina izvođenja spomenutih algoritama.

4.1. Opis skupa podataka korištenog za evaluaciju

Za potrebe evaluacije algoritama opisanih u trećem poglavlju potrebno je pronaći relevantne evaluacijske videozapise koji sadrže označene pješake. Odabrano je četiri evaluacijska videozapisa iz *MOT Benchmark 17 (Multiple Object Tracking Benchmark)* [23]. Ovaj skup podataka sadrži pedeset i dva videozapisa u raznim uvjetima s ručno označenim pješacima. Za potrebe evaluacije predloženog algoritma za detekciju i praćenje pješaka odabrana su po dva videozapisa sa statičnom kamerom i dva s kamerom u pokretu kako bi se testirao rad algoritama u raznim uvjetima. Razlučivost svih videozapisa jednaka je 1920x1080 piksela.

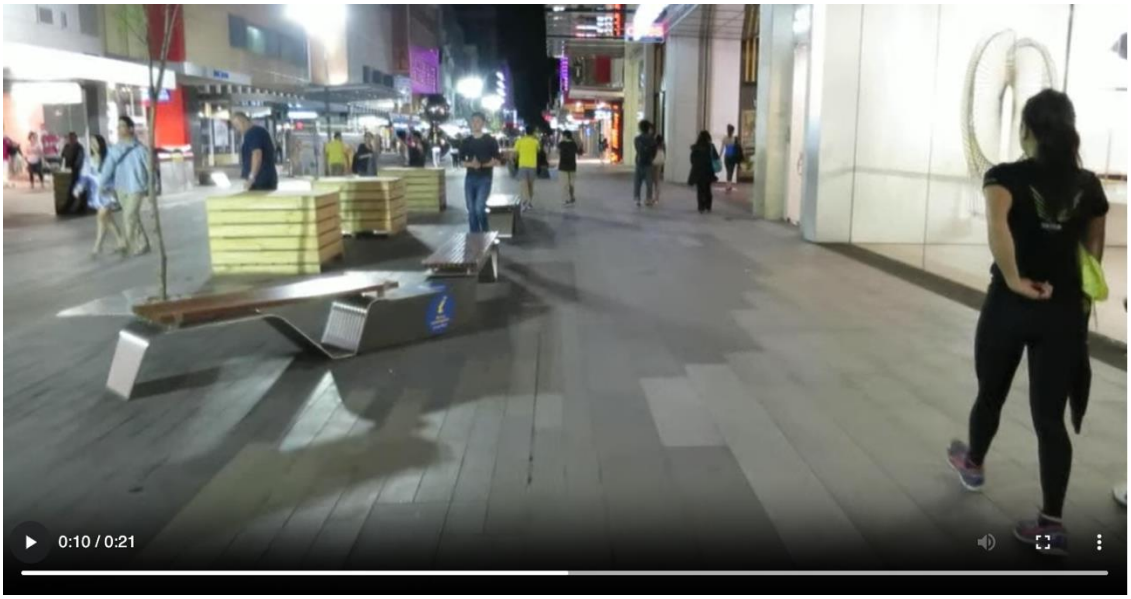
Prvi evaluacijski videozapis je *MOT17-02-SDP,* kamera je statična i prikazani su ljudi koji hodaju trgom u dnevnim uvjetima. Videozapis se sastoji od 600 označenih okvira. Isječak ovog videozapisa prikazan je na slici 4.1.

Sl. 4.1. Primjer okvira iz *MOT17-02-SDP* videozapisa

Drugi videozapis je *MOT17-09-SDP*, također sa statičnom kamerom iz niskog kuta snimanja. Prikazuje ljude koji hodaju ispred ulaza u zgradu. Sastoji se od 525 okvira s pripadnim oznakama. Isječak *MOT17-09-SDP* videozapisa je prikazan na slici 4.2.

Sl. 4.2. Primjer okvira iz videozapisa *MOT17-09-SDP*

Treći odabrani videozapis je *MOT17-10-SDP* koji se sastoji od 654 označena okvira. Prikazuje pješake noću i snimljen je iz kamere u pokretu. Primjer jednog okvira ovog videozapisa prikazan je na slici 4.3.



Sl. 4.3. Primjer okvira iz videozapisa *MOT17-10-SDP*

Posljednji videozapis korišten u svrhu evaluacije algoritama je *MOT17-13-SDP*. Ovaj videozapis se sastoji od 750 okvira i snimljen je s vrha autobusa na jednom raskrižju u dnevnim uvjetima. Mnogo pješaka se kreće pločnikom sa svih strana raskrižja. Većina pješaka u videozapisu je često zaklonjena drugim objektima. Primjer jednog okvira ovog videozapisa prikazan je na slici 4.4. Slika 4.5. prikazuje primjer pripadnih oznaka pješaka preuzetih u zasebnim tekstualnim datotekama.

Sl. 4.4 Primjer okvira iz videozapisa *MOT17-13-SDP*

```
1,1,130.0,225.0,51.0,131.0,1,1,1
2,1,131.0,224.5,51.0,131.5,1,1,1
3,1,132.0,224.5,51.0,131.5,1,1,1
4,1,133.0,224.0,51.0,132.0,1,1,1
5,1,134.0,224.0,51.0,132.0,1,1,1
6,1,135.0,224.0,51.0,132.0,1,1,1
7,1,136.0,224.0,51.5,132.0,1,1,1
8,1,137.0,224.0,51.5,132.0,1,1,1
9,1,138.0,224.0,52.0,132.0,1,1,1
```

Sl. 4.5. Primjer nekoliko originalnih oznaka za videozapis *MOT17-09-SDP*

Prva vrijednost predstavlja redni broj okvira videozapisa, druga vrijednost je identifikacijska oznaka pojedinog pješaka, zatim slijede *x* i *y* koordinate središta graničnog okvira, pa širina i visina graničnog okvira. Ostale vrijednosti nisu od značaja za evaluaciju predloženog algoritma za detekciju i praćenje pješaka. Zbog različitog načina zapisivanja graničnih okvira predloženog algoritma i zapisa graničnih okvira evaluacijskih videozapisa potrebno je skalirati vrijednosti. Budući da je razlučivost ovih videozapisa 1920x1080 piksela, *x* koordinata i širina su podijeljene

s 1920, a *y* koordinata i visina s 1080. Nakon ovog postupka primjer sa slike 4.5. pretvoren je u oblik kao na slici 4.6.

```
1,1,0.1354,0.4167,0.0531,0.2426,1,1,1
2,1,0.1365,0.4157,0.0531,0.2435,1,1,1
3,1,0.1375,0.4157,0.0531,0.2435,1,1,1
4,1,0.1385,0.4148,0.0531,0.2444,1,1,1
5,1,0.1395,0.4148,0.0531,0.2444,1,1,1
6,1,0.1406,0.4148,0.0531,0.2444,1,1,1
7,1,0.1417,0.4148,0.0536,0.2444,1,1,1
8,1,0.1427,0.4148,0.0536,0.2444,1,1,1
9,1,0.1437,0.4148,0.0541,0.2444,1,1,1
```

Sl. 4.6. Primjer nekoliko originalnih oznaka za videozapis *MOT17-09-SDP* nakon skaliranja

4.2. Opis mjera korištenih za evaluaciju

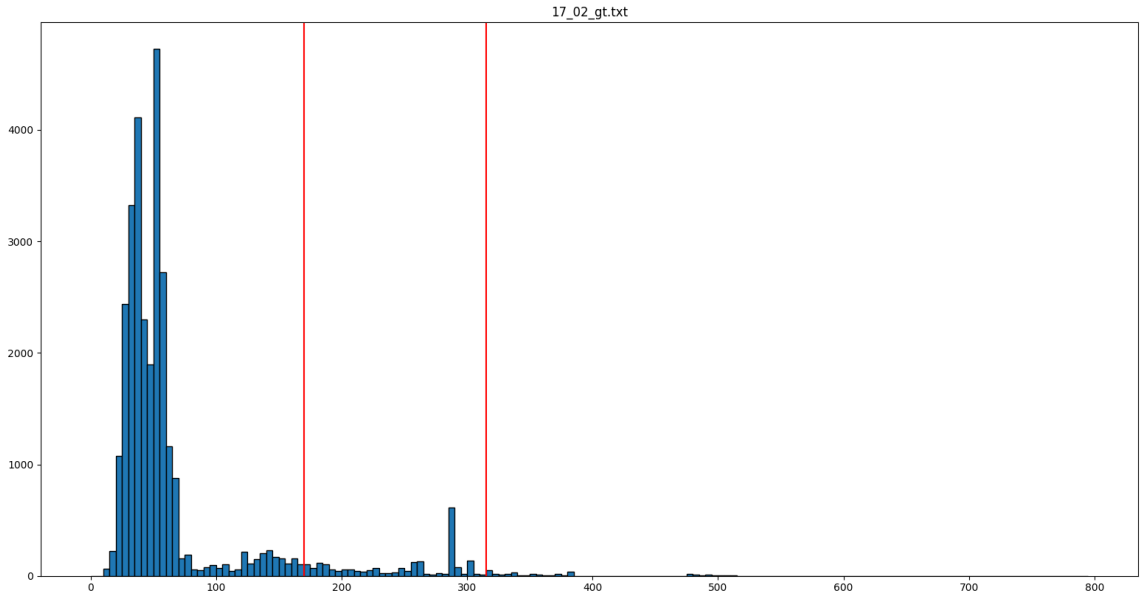
Za potrebe evaluacije rada algoritma za detekciju i praćenje pješaka korištene su mjere opisane u 2.1.1. U skupu podataka opisanom u dijelu 4.1. nalazi se mnogo označenih pješaka koji su vrlo mali i često detektori nisu u stanju prepoznati ih. Da bi se dobio detaljniji uvid u performanse rada algoritma mjera *AP* se promatra zasebno za male, srednje i velike objekte na testnim videozapisima. S obzirom na specifičnu situaciju koja se odnosi na zahtjeve ovoga rješenja i na vrstu skupa podataka na kojemu se vrši evaluacija, potrebno je odrediti vlastite granične vrijednosti za klasificiranje objekata u male, srednje i velike. Granične vrijednosti odnose se na visine pješaka označenih u videozapisu. Iz skupa korištenog za evaluaciju predloženog algoritma za detekciju i praćenje pješaka odabran je okvir videozapisa pomoću kojeg se može ilustrirati odabir vlastitih graničnih vrijednosti. Razmatrani su stvarni označeni podaci. Zatim su odabrana tri pješaka koji predstavljaju granične vrijednosti za klasifikaciju. Slika 4.7. prikazuje odabrani nasumični okvir s okruženim relevantnim graničnim pješacima i njihovim stvarnim oznakama. Pješak okružen zelenom krivuljom predstavlja prag za male objekte i njegova visina je 170 piksela. Pješak okružen žutom krivuljom predstavlja granicu između srednjih i velikih pješaka i njegova visina je 315 piksela. Prema ovim pragovima određeno je da pješaci niži od 170 piksela pripadaju u kategoriju mali (engl. *small*), pješaci visine između 171 i 315 piksela pripadaju u klasu srednji (engl. *medium*) i svi pješaci viši od 315 piksela pripadaju klasi veliki (engl. *large*).

Sl. 4.7. Nasumični okvir iz skupa za evaluaciju s ilustracijom klasifikacije po veličini

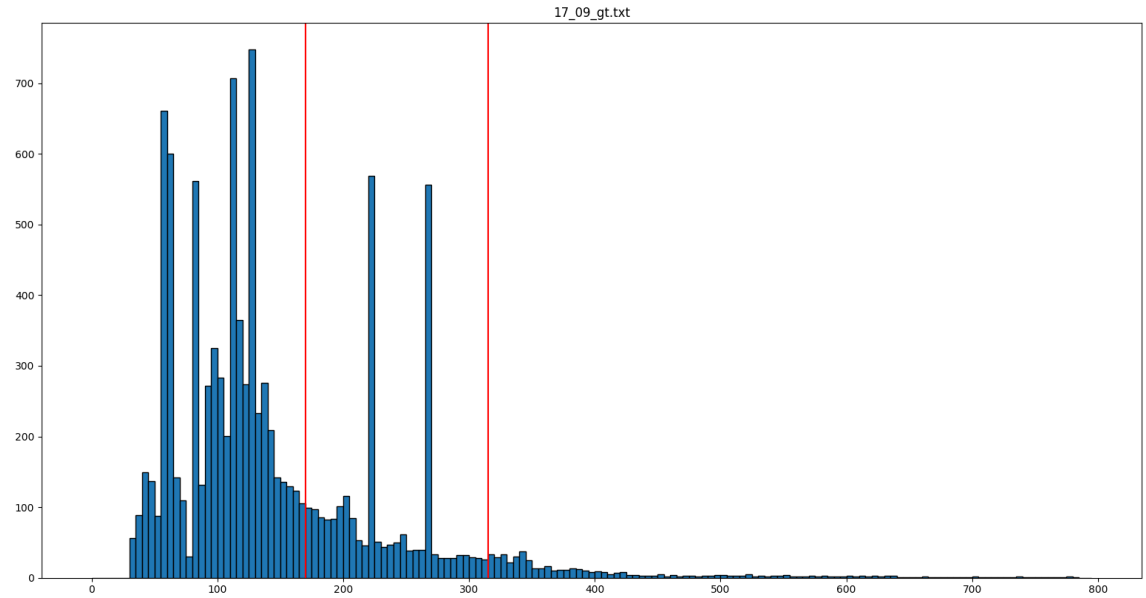
U tablici 4.1. se nalazi prikaz raspodjele graničnih okvira pješaka u kategorije mali, srednji i veliki. Slike 4.8., 4.9., 4.10. i 4.11. prikazuju histograme graničnih okvira s obzirom na visinu za evaluacijske videozapise. Korišten je razred od pet piksela. *X* os predstavlja visinu izraženu u pikselima, *y* os predstavlja broj graničnih okvira s pripadnom visinom. Svaki stupac predstavlja raspon od 5 piksela, crvene linije su vlastite granice koje definiraju što se smatra malim, srednjim i velikim pješakom.

Tablica 4.1. Raspodjela pješaka u kategorije za svaki videozapis

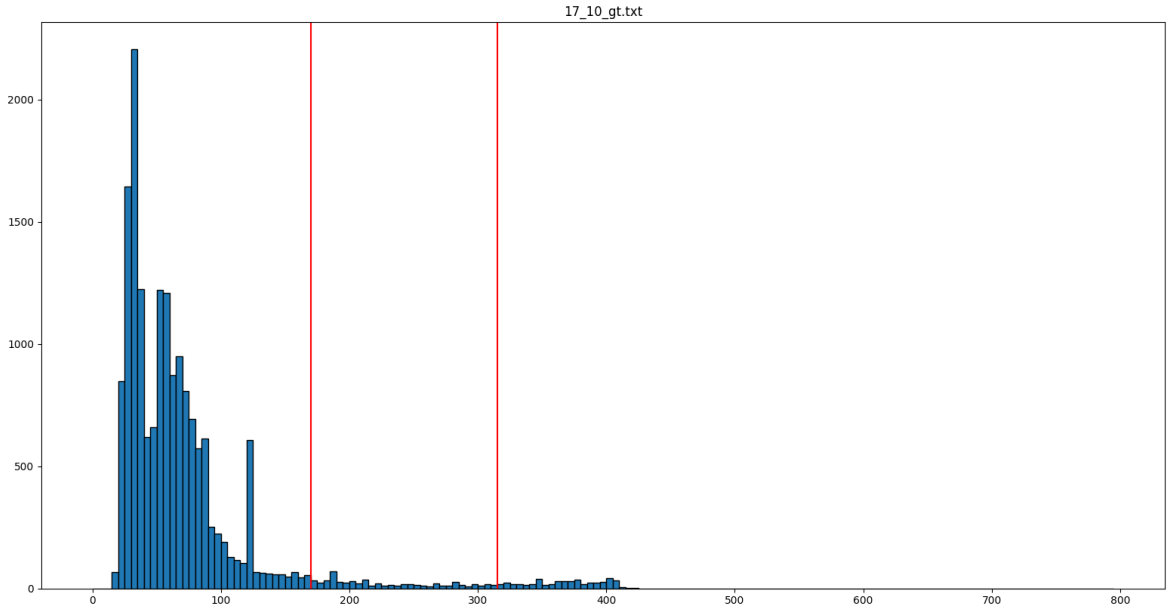
Videozapis	Mali	Srednji	Veliki
MOT17-02-DPM	27453	2234	315
MOT17-09-DPM	7284	2606	520
MOT17-10-DPM	16357	606	486
MOT17-13-DPM	19340	615	246
Ukupno	**70434**	**6061**	**1567**



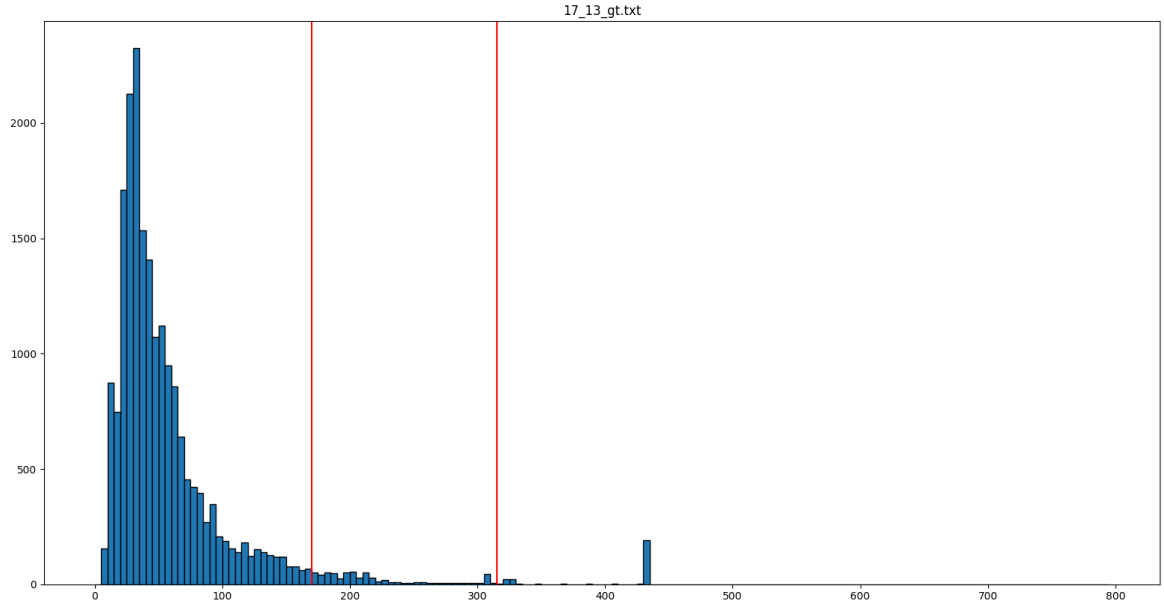
Sl. 4.8. Grafički prikaz distribucije graničnih okvira pješaka po visini za videozapis *MOT17-02-SDP*



Sl. 4.9. Grafički prikaz distribucije graničnih okvira pješaka po visini za videozapis *MOT17-09-SDP*



Sl. 4.10. Grafički prikaz distribucije graničnih okvira pješaka po visini za videozapis *MOT17-10-SDP*



Sl. 4.11. Grafički prikaz distribucije graničnih okvira pješaka po visini za videozapis *MOT17-13-SDP*

Iz ovih podataka je jasno vidljivo da se u skupu za evaluaciju nalazi velik broj oznaka pješaka u kategoriji mali. Kategoriji mali pripada 90% oznaka, kategoriji srednji 8% i kategoriji veliki 2% oznaka pješaka.

4.3. Rezultati evaluacije predloženog algoritma za detekciju i praćenje pješaka

Evaluacija algoritma je zasebno za predložene algoritme za detekciju i praćenje pješaka temeljene na *SORT* i *DeepSORT* algoritmu.

4.3.1. Evaluacija predloženog algoritma za detekciju i praćenje pješaka temeljena na *SORT* algoritmu

Za evaluaciju rješenja za detekciju i praćenje pješaka na osobnom računalu potrebno je rješenje pokrenuti na sva četiri videozapisa iz skupa za evaluaciju. Za rješenje sa SORT algoritmom pokrenuta su sva četiri videozapisa korištenjem četiri modela: *YOLOv7 original, YOLOv7 custom, YOLOv7 TL* i *YOLOv7 tiny TL* i to za argument *skip_frames* s vrijednostima 2, 5 i 10 što predstavlja broj okvira videozapisa između izvršavanja detekcije. Ovim načinom evaluacije rješenje sa *SORT* verzijom algoritma za praćenje potrebno je pokrenuti 48 puta. Rezultati rada predloženog algoritma za detekciju i praćenje pješaka zapisani su na lokalno računalo u obliku videozapisa i tekstualnih datoteka sa zapisima graničnih okvira. Usporedbom ovih zapisa i stvarnih oznaka računaju se mjere: preciznost, odziv i *F1* ocjena. Tablice 4.2., 4.3., 4.4. i 4.5. prikazuju postignute rezultate preciznosti, odziva i *F1* ocjene za svaki videozapis zasebno. U svakom retku narednih tablica nalaze se rezultati za svaki od izgrađenih detektora: *original* za originalni *YOLOv7* detektor, *custom* za detektor treniran bez pretreniranih težina, *TL* za detektor treniran metodom prijenosnog učenja i *tiny TL* za *YOLOv7 tiny* detektor treniran metodom prijenosnog učenja. Prva podtablica svake od ovih tablica predstavlja rezultate kada se detekcija vrši svaki 2. okvir, druga podtablica predstavlja rezultate kada se detekcija vrši svaki 5. okvir i treća podtablica predstavlja rezultate kada se detekcija vrši svaki 10. okvir.

Tablica 4.2. Preciznost, odziv i *F1* ocjena *SORT* verzije algoritma za videozapis *MOT17-02-SDP*

Model	***P***	***R***	***F1***
original	0,7560	0,2863	0,4154
custom	0,8260	0,1870	0,3050
TL	0,8060	0,1704	0,2813
tiny TL	0,8020	0,1341	0,2230

Model	***P***	***R***	***F1***
original	0,7595	0,2619	0,3895
custom	0,8573	0,1745	0,2900
TL	0,8260	0,1559	0,2624
tiny TL	0,8040	0,1240	0,2148

Model	***P***	***R***	***F1***
original	0,7576	0,2242	0,3460
custom	0,8394	0,1462	0,2490
TL	0,8225	0,1332	0,2292
tiny TL	0,7797	0,1063	0,1871

Tablica 4.3. Preciznost, odziv i *F1* ocjena *SORT* verzije algoritma za videozapis *MOT17-09-SDP*

Model	***P***	***R***	***F1***
original	0,7018	0,4858	0,5742
custom	0,9333	0,4002	0,5602
TL	0,9207	0,4353	0,5911
tiny TL	0,7986	0,3998	0,5328

Model	***P***	***R***	***F1***
original	0,7341	0,4610	0,5664
custom	0,9020	0,3572	0,5120
TL	0,9132	0,4032	0,5594
tiny TL	0,7862	0,3635	0,4971

Model	***P***	***R***	***F1***
original	0,6865	0,3793	0,4886
custom	0,8158	0,2846	0,4220
TL	0,8253	0,3244	0,4657
tiny TL	0,7100	0,2835	0,4035

Tablica 4.4. Preciznost, odziv i *F1* ocjena *SORT* verzije algoritma za videozapis *MOT17-10-SDP*

Model	***P***	***R***	***F1***
original	0,7385	0,4953	0,5929
custom	0,6551	0,5040	0,5696
TL	0,6999	0,4767	0,5671
tiny TL	0,7217	0,2740	0,3973

Model	***P***	***R***	***F1***
original	0,6500	0,4223	0,5120
custom	0,5966	0,4200	0,4930
TL	0,6466	0,3850	0,4826
tiny TL	0,6343	0,2284	0,3359

Model	***P***	***R***	***F1***
original	0,6091	0,3182	0,4180
custom	0,5876	0,3134	0,4088
TL	0,6120	0,2863	0,3900
tiny TL	0,5588	0,1631	0,2525

Tablica 4.5. Preciznost, odziv i *F1* ocjena *SORT* verzije algoritma za videozapis *MOT17-13-SDP*

Model	*P*	*R*	*F1*
original	0,6324	0,3366	0,4393
custom	0,5682	0,2300	0,3274
TL	0,6270	0,1993	0,3024
tiny TL	0,5100	0,1264	0,2025

Model	*P*	*R*	*F1*
original	0,5016	0,2453	0,3294
custom	0,4692	0,1547	0,2327
TL	0,5051	0,1383	0,2172
tiny TL	0,3792	0,0850	0,1390

Model	*P*	*R*	*F1*
original	0,4679	0,1527	0,2303
custom	0,4637	0,1017	0,1669
TL	0,5024	0,0871	0,1485
tiny TL	0,3357	0,0518	0,0905

Iz rezultata evaluacije vidljivo je da predloženi algoritam za detekciju i praćenje pješaka postiže najbolje rezultate svih triju metrika s izvršavanjem detekcije svaki drugi okvir videozapisa, a najlošije rezultate s izvršavanjem detekcije svaki deseti okvir videozapisa što je sukladno očekivanju. Gledajući *F1* ocjenu kao najrelevantniju mjeru za ocjenu općeg rada pojedinog detektora, vidljivo je da originalni detektor postiže najbolje rezultate a *YOLOv7 tiny* najlošije. Možemo zaključiti da su se trenirani detektori prilagodili na trening podatke i imaju nešto lošije performanse u evaluacijskim videozapisima koji se razlikuju od trening podataka. Također možemo primijetiti da je općenito preciznost treniranih detektora porasla, a odziv se smanjio u odnosu na originalni detektor.

Za svaku klasu po veličini zasebno je izračunata mjera AP@0.5 i AP@0.5:0.95. Tablice 4.6., 4.7., 4.8. i 4.9. Prikazuju postignutu prosječnu preciznost za svaki evaluacijski videozapis zasebno. U svakoj od ovih tablica *APs (AP small)* predstavlja prosječnu preciznost za male, *APm (AP medium*) za srednje i *APl (AP large*) za velike pješake.

Tablica 4.6. Rezultati prosječne preciznosti za male, srednje i velike objekte za videozapis *MOT17-02-SDP*

	Detekcija svaki 2. okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,1388	0,1868	0,3941	0,4786	0,6060	0,7596
custom	0,0944	0,1232	0,3217	0,4014	0,4665	0,6221
TL	0,0748	0,0941	0,3543	0,4372	0,3247	0,4531
tiny TL	0,0712	0,0956	0,3313	0,4178	0,2698	0,3811
	Detekcija svaki 5. Okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,1272	0,1726	0,3730	0,4694	0,4989	0,622
custom	0,0874	0,1143	0,3020	0,3846	0,4255	0,5569
TL	0,0675	0,0854	0,3375	0,4239	0,2873	0,4026
tiny TL	0,0677	0,0900	0,3192	0,4084	0,2316	0,3226
	Detekcija svaki 10. okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,1057	0,1447	0,3100	0,3975	0,3940	0,5060
custom	0,0729	0,0979	0,2713	0,3511	0,3362	0,4436
TL	0,0578	0,0745	0,2989	0,3904	0,2273	0,3240
tiny TL	0,0580	0,0792	0,2823	0,3715	0,1167	0,1707

Tablica 4.7. Rezultati prosječne preciznosti za male, srednje i velike objekte za videozapis *MOT17-09-SDP*

	Detekcija svaki 2. okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,2478	0,3297	0,3022	0,3742	0,4340	0,5589
custom	0,2319	0,3076	0,3030	0,3866	0,4349	0,5842
TL	0,2869	0,3684	0,2951	0,3689	0,4100	0,5477
tiny TL	0,1500	0,1987	0,2890	0,3684	0,4104	0,5484
	Detekcija svaki 5. Okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,2575	0,3462	0,2550	0,3267	0,3498	0,4603
custom	0,1974	0,2630	0,2430	0,3158	0,3924	0,5282
TL	0,2523	0,3219	0,2759	0,3464	0,3831	0,5136
tiny TL	0,1305	0,1761	0,2547	0,3279	0,3523	0,4750
	Detekcija svaki 10. okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,2116	0,2901	0,1778	0,2394	0,3273	0,4282
custom	0,1563	0,2150	0,1818	0,2470	0,2937	0,3978
TL	0,1970	0,2599	0,1919	0,2599	0,2909	0,3932
tiny TL	0,0914	0,1260	0,1579	0,2173	0,2779	0,3722

Tablica 4.8. Rezultati prosječne preciznosti za male, srednje i velike objekte za videozapis *MOT17-10-SDP*

	Detekcija svaki 2. okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,2108	0,2918	0,3745	0,4642	0,5341	0,6548
custom	0,1855	0,2575	0,3951	0,4986	0,4671	0,5913
TL	0,1854	0,2581	0,4124	0,5275	0,4603	0,5834
tiny TL	0,1448	0,2043	0,3140	0,3925	0,4537	0,5891
	Detekcija svaki 5. Okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,1511	0,2138	0,3522	0,4595	0,5003	0,6274
custom	0,1359	0,1928	0,3599	0,4728	0,4670	0,5966
TL	0,1338	0,1895	0,3726	0,4818	0,4860	0,6282
tiny TL	0,1057	0,1511	0,3315	0,4325	0,4431	0,5827
	Detekcija svaki 10. okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,1021	0,1476	0,2502	0,3382	0,4145	0,5554
custom	0,0941	0,1354	0,2909	0,3925	0,4515	0,6027
TL	0,0884	0,1271	0,2799	0,3807	0,4080	0,5483
tiny TL	0,0696	0,1005	0,2930	0,4013	0,4273	0,5833

Tablica 4.9. Rezultati prosječne preciznosti za male, srednje i velike objekte za videozapis *MOT17-13-SDP*

	Detekcija svaki 2. okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,1153	0,1598	0,0028	0,0037	0	0
custom	0,0708	0,1003	0,0018	0,0025	0	0
TL	0,0687	0,0962	0,0017	0,0022	0	0
tiny TL	0,0442	0,0632	0,0038	0,0054	0	0
	Detekcija svaki 5. Okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,0667	0,0957	0,0006	0	0	0
custom	0,0403	0,0583	0	0	0	0
TL	0,0378	0,0545	0	0	0	0
tiny TL	0,0280	0,0408	0	0	0	0
	Detekcija svaki 10. okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,0394	0,0575	0	0	0	0
custom	0,0270	0,0398	0	0	0	0
TL	0,0236	0,0345	0	0	0	0
tiny TL	0,0191	0,0282	0	0	0	0

Sukladno očekivanjima, prosječna preciznost je najveća za granične okvire iz kategorije veliki, a najmanja za granične okvire iz kategorije mali. Predloženi algoritam za detekciju i praćenje pješaka temeljen na *SORT* algoritmu za praćenje postiže vrlo dobre rezultate za mjeru *AP@*0.5 uz izvođenje sa zadovoljavajućim brojem okvira po sekundi. U tablici 4.9. je vidljivo da prosječna preciznost u nekim slučajevima iznosi 0. Razlog tome je taj što se radi o videozapisu s kamerom u pokretu. Također, u spomenutom videozapisu ima jako malo oznaka pješaka svrstanih u kategorije srednji i veliki. Uz sve spomenuto, detekcija svaki 5. ili 10. okvir nije dovoljna da bi algoritam uspješno pratio pješake iz spomenutih kategorija u ovome videozapisu.

4.3.2. Evaluacija predloženog algoritma za detekciju i praćenje pješaka temeljena na *DeepSORT* algoritmu za praćenje

Evaluacija predloženog algoritma za detekciju i praćenje pješaka temeljenog na *DeepSORT* algoritmu izvršava se kao i evaluacija provedena u potpoglavlju 4.3.1. Tablice 4.10., 4.11., 4.12. i 4.13. redom predstavljaju rezultate preciznosti, odziva i *F1* ocjene za videozapise *MOT17-02-SDP, MOT17-09-SDP, MOT17-10-SDP i MOT17-13-SDP*.

Tablica 4.10. Preciznost, odziv i *F1* ocjena Deep*SORT* verzije algoritma za videozapis *MOT17-02-SDP*

Model	***P***	***R***	***F1***
original	0,8206	0,2393	0,3706
custom	0,9501	0,1394	0,2432
TL	0,8511	0,1287	0,2236
tiny TL	0,9797	0,1209	0,2153

Model	***P***	***R***	***F1***
original	0,7886	0,2302	0,3564
custom	0,9248	0,1348	0,2353
TL	0,8403	0,1253	0,2182
tiny TL	0,9751	0,0969	0,1765

Model	***P***	***R***	***F1***
original	0,7232	0,2086	0,3239
custom	0,9072	0,1274	0,2235
TL	0,8218	0,1183	0,2068
tiny TL	0,9220	0,0670	0.1249

Tablica 4.11. Preciznost, odziv i *F1* ocjena Deep*SORT* verzije algoritma za videozapis *MOT17-09-SDP*

Model	***P***	***R***	***F1***
original	0,8668	0,4522	0,5943
custom	0,9798	0,2849	0,4414
TL	0,9665	0,2969	0,4543
tiny TL	0,9045	0,3446	0,499

Model	***P***	***R***	***F1***
original	0,8111	0,4151	0,5491
custom	0,9255	0,2637	0,4104
TL	0,9313	0,2797	0,4302
tiny TL	0,8227	0,2447	0,3773

Model	***P***	***R***	***F1***
original	0,7509	0,3824	0,5067
custom	0,7965	0,2292	0,3559
TL	0,8046	0,2369	0,3659
tiny TL	0,7596	0,1832	0,2952

Tablica 4.12. Preciznost, odziv i *F1* ocjena Deep*SORT* verzije algoritma za videozapis *MOT17-10-SDP*

Model	***P***	***R***	***F1***
original	0,9043	0,4501	0,6011
custom	0,8907	0,4398	0,5889
TL	0,9624	0,3747	0,5374
tiny TL	0,8941	0,2356	0,3729

Model	***P***	***R***	***F1***
original	0,7694	0,3821	0,5106
custom	0,7554	0,3694	0,4962
TL	0,8294	0,3212	0,4631
tiny TL	0,8108	0,1596	0,2667

Model	***P***	***R***	***F1***
original	0,5989	0,2962	0,3964
custom	0,6042	0,2951	0,3964
TL	0,6658	0,2541	0,3678
tiny TL	0,6878	0,0866	0,1538

Tablica 4.13. Preciznost, odziv i *F1* ocjena Deep*SORT* verzije algoritma za videozapis *MOT17-13-SDP*

Model	***P***	***R***	***F1***
original	0,9411	0,2782	0,4295
custom	0,9056	0,1851	0,3072
TL	0,9771	0,1763	0,2988
tiny TL	0,900	0,0620	0,1158

Model	***P***	***R***	***F1***
original	0,7157	0,2102	0,3249
custom	0,7417	0,1464	0,2446
TL	0,7071	0,1219	0,2081
tiny TL	0,696	0,0297	0,0570

Model	***P***	***R***	***F1***
original	0,4632	0,1346	0,2086
custom	0,5114	0,1019	0,1699
TL	0,5364	0,0898	0,1538
tiny TL	0,3649	0,0061	0,0120

Iz rezultata provedene evaluacije je vidljivo da predloženi algoritam za detekciju i praćenje pješaka postiže vrlo dobre mjere preciznost i *F1* ocjena uz nešto lošiju mjeru odziva. Izravan razlog ovakvog odziva je taj što detektori ne mogu detektirati velik broj prisutnih pješaka u evaluacijskim videozapisima jer su mali ili zaklonjeni. Zaključak je isti kao i kod analize evaluacije iz potpoglavlja 4.3.1. Originalni detektor općenito postiže najbolji rezultat za *F1* ocjenu. U odnosu na originalni detektor, trenirani detektori većinom postižu bolju preciznost i lošiji odziv. *YOLOv7 tiny* očekivano postiže najlošije rezultate zbog svoje jednostavnije arhitekture.

Tablice 4.14., 4.15., 4.16. i 4.17. prikazuju rezultat prosječne preciznos*ti* za evaluacijske videozapise redom: *MOT17-02-SDP*, *MOT17-09-SDP*, *MOT17-10-SDP* i *MOT17-13-SDP*.

Tablica 4.14. Rezultati prosječne preciznosti za male, srednje i velike objekte za videozapis *MOT17-02-SDP*

	Detekcija svaki 2. okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,1193	0,1603	0,4076	0,5005	0,6263	0,8009
custom	0,0706	0,0893	0,3133	0,3845	0,458	0,6204
TL	0,0512	0,0637	0,3317	0,4095	0,4155	0,5612
tiny TL	0,0559	0,0727	0,3358	0,4236	0,3147	0,4351
	Detekcija svaki 5. Okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,114	0,1546	0,3719	0,4712	0,5437	0,716
custom	0,0677	0,0869	0,2984	0,3733	0,4255	0,5764
TL	0,0489	0,0617	0,3171	0,3968	0,4181	0,582
tiny TL	0,0534	0,0699	0,3212	0,416	0,1898	0,2716
	Detekcija svaki 10. okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,0992	0,1387	0,344	0,4531	0,4508	0,5921
custom	0,0604	0,0797	0,2804	0,3561	0,3458	0,471
TL	0,0424	0,0551	0,3008	0,3816	0,2705	0,367
tiny TL	0,0481	0,064	0,301	0,3974	0,105	0,1508

Tablica 4.15. Rezultati prosječne preciznosti za male, srednje i velike objekte za videozapis *MOT17-09-SDP*

	Detekcija svaki 2. okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,2969	0,3937	0,309	0,3827	0,4064	0,53
custom	0,1445	0,1895	0,2595	0,3255	0,3702	0,4972
TL	0,1576	0,1999	0,2602	0,3222	0,3589	0,4814
tiny TL	0,0922	0,1195	0,232	0,2937	0,3141	0,4137
	Detekcija svaki 5. Okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,277	0,3741	0,2529	0,3369	0,3386	0,4402
custom	0,133	0,1767	0,229	0,3047	0,3033	0,4126
TL	0,1426	0,1825	0,233	0,3018	0,3055	0,4142
tiny TL	0,0811	0,1086	0,2102	0,2748	0,26	0,3534
	Detekcija svaki 10. okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,2362	0,3214	0,2039	0,2847	0,2529	0,3474
custom	0,1112	0,1517	0,1591	0,2191	0,2413	0,3416
TL	0,1193	0,1607	0,1679	0,2338	0,2433	0,3357
tiny TL	0,0739	0,0998	0,1678	0,2315	0,2361	0,3288

Tablica 4.16. Rezultati prosječne preciznosti za male, srednje i velike objekte za videozapis *MOT17-10-SDP*

	Detekcija svaki 2. okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,2382	0,3279	0,4343	0,5572	0,5468	0,6882
custom	0,2299	0,316	0,4146	0,5341	0,5238	0,6658
TL	0,2073	0,2815	0,3919	0,51	0,552	0,6972
tiny TL	0,1705	0,2366	0,3496	0,4504	0,5369	0,7
	Detekcija svaki 5. Okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,1609	0,2296	0,3729	0,5048	0,485	0,6405
custom	0,1534	0,218	0,357	0,4826	0,4366	0,5762
TL	0,1421	0,2006	0,3467	0,4777	0,5185	0,6793
tiny TL	0,109	0,1562	0,3078	0,406	0,4621	0,6186
	Detekcija svaki 10. okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,0921	0,133	0,2156	0,3003	0,3493	0,4958
custom	0,0922	0,1327	0,2504	0,3447	0,3501	0,5017
TL	0,0851	0,1228	0,2332	0,33	0,4645	0,623
tiny TL	0,0706	0,1022	0,2573	0,3523	0,429	0,5806

Tablica 4.17. Rezultati prosječne preciznosti za male, srednje i velike objekte za videozapis *MOT17-13-SDP*

	Detekcija svaki 2. okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,2969	0,3937	0,309	0,3827	0,4064	0,53
custom	0,1445	0,1895	0,2595	0,3255	0,3702	0,4972
TL	0,1576	0,1999	0,2602	0,3222	0,3589	0,4814
tiny TL	0,0636	0,0902	0,0006	0,0009	0	0
	Detekcija svaki 5. Okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,277	0,3741	0,2529	0,3369	0,3386	0,4402
custom	0,133	0,1767	0,229	0,3047	0,3033	0,4126
TL	0,1426	0,1825	0,233	0,3018	0,3055	0,4142
tiny TL	0,0403	0,0587	0	0	0	0
	Detekcija svaki 10. okvir:					
	APs		***APm***		***APl***	
	0.50:0.95	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***	***0.50:0.95***	***IoU=0.50***
original	0,2362	0,3214	0,2039	0,2847	0,2529	0,3474
custom	0,1112	0,1517	0,1591	0,2191	0,2413	0,3416
TL	0,1193	0,1607	0,1679	0,2338	0,2433	0,3357
tiny TL	0,0204	0,0301	0	0	0	0

Iz rezultata evaluacije vidljivo je da predloženi algoritam za detekciju i praćenje pješaka temeljen na *DeepSORT* algoritmu postiže vrlo dobru prosječnu preciznost za granične okvire koji

pripadaju kategoriji veliki i nešto manju za one koji pripadaju kategoriji mali. Osvrtom na rezultate evaluacije u potpoglavlju 4.3.1. vidljivo je da predloženi algoritam za detekciju i praćenje pješaka temeljen na *DeepSORT* verziji algoritma postiže bolju prosječnu preciznost od onoga temeljenog na *SORT* algoritmu za praćenje. U oba slučaja su rezultati *APs* vrlo loši jer detektori nisu u mogućnosti prepoznati pješake s oznakama iz te kategorije.

4.3.3. Evaluacija predloženog algoritma za detekciju i praćenje pješaka s obzirom na brzinu izvođenja

Za potrebe evaluacije predloženog algoritma za detekciju i praćenje pješaka mjereno je i vrijeme izvođenja algoritma. Algoritmi s verzijama detektora*: original, custom* i *TL* pokrenuti su na računalu s *Intel Core i7-7700* procesorom i 16GB *RAM* memorije, a algoritam s *tiny* verzijom *YOLOv7* detektora pokrenut je na *NVIDIA Jetson Nano* platformi ograničenih resursa s *ARM Cortex (ARMv8-A, Quad core)* procesorom, integriranom *NVIDIA Tegra X1* grafičkom karticom i 4GB *RAM* memorije. Prilikom pokretanja svakog od evaluacijskih videozapisa ranije navedenim argumentima, mjereno je vrijeme izvođenja pojedinog videozapisa. Dijeljenjem broja okvira svakog videozapisa s izmjerenim vremenom izvođenja dobiva se prosječni broj okvira po sekundi.

Tablica 4.18. prikazuje rezultate evaluacije za brzinu izvođenja predloženog algoritma za detekciju i praćenje temeljenog na *SORT* algoritmu. Prva podtablica sadrži brojeve okvira po sekundi za kada se detekcija izvršava svaki 2. okvir evaluacijskog videozapisa, druga podtablica kada se detekcija izvršava svaki 5. okvir i treća podtablica sadrži brojeve okvira po sekundi kada se detekcija izvršava svaki 10. okvir videozapisa.

Tablica 4.18. Broj okvira po sekundi predloženog algoritma za detekciju i praćenje pješaka temeljenog na *SORT* algoritmu za pojedini evaluacijski videozapis

Videozapis	***MOT17-02-SDP***	***MOT17-09-SDP***	***MOT17-10-SDP***	***MOT17-13-SDP***
original	2,61	2,57	2,58	2,53
custom	3,87	3,77	3,82	3,86
TL	3,92	3,89	3,74	3,83
tiny TL	7,5	5,65	6,67	7,28

Videozapis	***MOT17-02-SDP***	***MOT17-09-SDP***	***MOT17-10-SDP***	***MOT17-13-SDP***
original	5,77	6,32	6	6,15
custom	8,33	8,47	8,72	8,15
TL	8,95	8,75	8,72	8,33
tiny TL	11,32	9,21	9,48	11,9

Videozapis	***MOT17-02-SDP***	***MOT17-09-SDP***	***MOT17-10-SDP***	***MOT17-13-SDP***
original	11,32	11,17	10,72	10,71
custom	15,38	16,4	12,34	14,15
TL	13,95	15,91	13,91	15,96
tiny TL	15,29	14,19	12,82	15,3

Vidljivo je kako predloženi algoritam za detekciju i praćenje pješaka s originalnim detektorom postiže manje okvira po sekundi od ostalih. Razlog tome je što spomenuti algoritam ima najveći odziv kao što je vidljivo u rezultatima evaluacije u potpoglavljima 4.3.1. i 4.3.2. Većim brojem praćenih objekata raste i vrijeme izvođenja predloženog algoritma. Predloženi algoritam za detekciju i praćenje pješaka temeljen na *YOLOv7 tiny* detektoru se izvodi najbrže zbog jednostavnije arhitekture mreže. Ova verzija se izvodi dosta brže od ostalih unatoč tome što je pokrenuta na platformi *NVIDIA Jetson Nano* koja raspolaže ograničenim resursima.

Tablica 4.19. prikazuje rezultate evaluacije za brzinu izvođenja predloženog algoritma za detekciju i praćenje temeljenog na *DeepSORT* algoritmu. Prva podtablica sadrži brojeve okvira po sekundi za kada se detekcija izvršava svaki 2. okvir evaluacijskog videozapisa, druga podtablica kada se detekcija izvršava svaki 5. okvir i treća podtablica sadšži brojeve okvira po sekundi kada se detekcija izvršava svaki 10. okvir videozapisa.

Tablica 4.19. Broj okvira po sekundi predloženog algoritma za detekciju i praćenje pješaka temeljenog na *DeepSORT* algoritmu za pojedini evaluacijski videozapis

Videozapis	***MOT17-02-SDP***	***MOT17-09-SDP***	***MOT17-10-SDP***	***MOT17-13-SDP***
original	1,6	1,65	1,63	1,68
custom	2,56	2,59	2,44	2,54
TL	2,47	2,54	2,45	2,6
tiny TL	4,2	4,44	4,39	4,57

Videozapis	***MOT17-02-SDP***	***MOT17-09-SDP***	***MOT17-10-SDP***	***MOT17-13-SDP***
original	3,12	3,18	2,98	3,22
custom	4,76	4,78	4,75	4,47
TL	5,17	5,06	5,36	5,36
tiny TL	6,74	6,54	6,93	6,67

Videozapis	***MOT17-02-SDP***	***MOT17-09-SDP***	***MOT17-10-SDP***	***MOT17-13-SDP***
original	3,89	3,86	3,65	3,98
custom	6,56	6,07	6,82	6,4
TL	5,83	5,87	5,52	6,1
tiny TL	5,91	5,74	5,43	5,97

Iz rezultata evaluacije predloženog algoritma za detekciju i praćenje pješaka vidljivo je da se algoritam temeljen na *SORT* algoritmu za praćenje izvodi višestruko brže u odnosu na onaj temeljen na *DeepSORT* algoritmu za praćenje.

Predloženi algoritam za detekciju i praćenje se izvodi u prosjeku 50% brže za detekciju svaki drugi okvir, 80% brže za detekciju svaki peti okvir i u prosjeku 90% brže za detekciju svaki deseti okvir u odnosu na algoritam koji ne koristi praćenje pješaka. Postignuto je da se rješenje sa *SORT* algoritmom izvodi brzinom od petnaest okvira po sekundi na ugradbenoj platformi koja ima

ograničene resurse uz naglasak da je na testnim videozapisima mnogo pješaka koje rješenje prepoznaje što je dodatno kompleksno za izvođenje. Ovakva detaljna evaluacija dala je na uvid prednosti i nedostatke pristupa ovoga rješenja. Potrebno je naći optimalan omjer brzine izvođenja i performansi rada rješenja. Dodatnom optimizacijom algoritma i eventualnim dodatnim treningom detektora može se dodatno unaprijediti ovo rješenje.

5. ZAKLJUČAK

U ovom radu razvijen je algoritam za detekciju i praćenje pješaka temeljen na *YOLOv7* detektoru i *SORT* i *DeepSORT* algoritmima za praćenje. Detektor je treniran na vlastitom skupu podataka i kombiniran je s algoritmima za praćenje. Razvijeno rješenje je detaljno evaluirano na evaluacijskim videozapisima i implementirano je na ugradbenu platformu *NVIDIA Jetson Nano* uz korištenje *YOLOv7 tiny* detektora. Budući da se radi o vrlo kompleksnom problemu, postignuti su vrlo dobri rezultati za detekciju i praćenje. Detektor je treniran na velikom skupu podataka koji se sastoji od slika pješaka označenih graničnim okvirima. U odnosu na originalni detektor, trenirani detektori postižu bolje performanse na testnom dijelu vlastitog skupa, a na evaluacijskim videozapisima originalni detektor ima bolji rezultat za odziv i *F1* ocjenu. Predloženi algoritam za detekciju i praćenje pješaka koristi kombinaciju detektora i algoritma za praćenje pa su postignuta dodatna poboljšanja u smislu brzine izvođenja algoritma i praćenja pješaka. Dodatnim treniranjem detektora i optimizacijom algoritma moguće je dobiti još bolje rezultate za detekciju i praćenje pješaka.

LITERATURA

[1] J. Redmon, S. Divvala, R. Girshick, A. Farhadi „*You Only Look Once: Unified, Real-Time Object Detection*“, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016. str. 779–88.
[2] A. Bewley, Z. Ge, L. Ott, F. Ramos, B. Upcroft, „*Simple Online and Realtime Tracking*“, 2016, IEEE International Conference on Image Processing (ICIP); Phoenix, AZ. IEEE; 2016. str. 3463-8.
[3] N. Wojke, A. Bewley, D. Paulus, „*Simple online and realtime tracking with a deep association metric*“, *2017*, IEEE International Conference on Image Processing (ICIP); 2017 Bejing, China. IEEE; 2017. str. 3645-9.
[4] Y. Pang, M. Sun, X. Jiang, X. Li., „*Convolution in Convolution for Network in Network*“, Sv. 29, IEEE Transactions on Neural Networks and Learning Systems Institute of Electrical and Electronics Engineers (IEEE); 2018. str. 1587–97.
[5] J. Redmon, A. Farhadi, *„YOLO9000: Better, Faster, Stronger*“, University of Washington, 2016.
[6] J. Redmon, A. Farhadi, „*YOLOv3: An Incremental Improvement*“, University of Washington, 2018.
[7] T.Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, „*Feature pyramid networks for object detection*“, 2016.
[8] A. Bochkovskiy, C. Wang, H. M. Liao, „*YOLOv4: Optimal Speed and Accuracy of Object Detection“,* 2020, Institute of Information Science Academia Sinica, Taiwan.
[9] YOLOv5, dostupno na: https://github.com/ultralytics/yolov5/ [14.7.2024.]
[10] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei, „*YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications*“, 2022.
[11] C. Wang, A. Bochkovskiy, H. M. Liao, „*YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*“, 2022.
[12] Y. Sun, Y. Yan, J.Zhao, C, Cai, „*Research on Vision-based pedestrian detection and tracking algorithm*“, IEEE International Conference on Mechatronics and Automation, 2022.
[13] A. Kumar, T. Singh, P. Duraisamy, „*Detection And Tracking of Multiple Pedestrians Using Deep Learning*“, International Conference on Computing Communication and Networking Technologies, 2023.
[14] S. Jiang, W.Li, J.Zhou, „*Pedestrian Target Tracking Algorithm on Fusion Detection*“, IEEE International Conference on Networking, Sensing and Control, 2022.
[15] J. Qu, S. Zhang, „*Research on Video Tracking Algorithm Based on Yolo Target Detection*“, 6th International Conference on Computer Network, Electronic and Automation, 2023.
[16] Y. Gong, J. Chi, X. Yu, C. Wu and Z. Jia, "*A Modified Multi-Pedestrian Tracking System*", Chinese Control Conference, China, 2019
[17] Jetson Nano, dostupno na: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development/, [14.7.2024].
[18] KITTI skup podataka, dostupno na: https://www.cvlibs.net/datasets/kitti/, [2.2.2024.].
[19] INRIA skup podataka, dostupno na: https://universe.roboflow.com/pascal-to-yolo-8yygq/inria-person-detection-dataset, [2.2.2024.].
[20] EuroCity persons skup podataka, dostupno na: https://eurocity-dataset.tudelft.nl/eval/overview/statistics, [2.2.2024.].
[21] YOLOv7 GitHub, dostupno na: https://github.com/WongKinYiu/yolov7, [14.5.2024.].
[22] SORT GitHub, dostupno na: https://github.com/abewley/sort, [14.5.2024.].
[23] MOT Challenge, dostupno na: https://motchallenge.net/data/MOT17/ [26.2.2024.].

SAŽETAK

U ovom radu opisan je razvoj rješenja za detekciju i praćenje pješaka u okviru autonomne vožnje. Prvo je objašnjen moderan način detekcije i praćenja pješaka u digitalnim slikama. Nakon toga je opisano nekoliko postojećih rješenja za detekciju i praćenje objekata. Rješenje se temelji na *YOLOv7* algoritmu za detekciju i *SORT* i *DeepSORT* algoritmima za praćenje. Algoritam za detekciju treniran je na vlastitom skupu podataka koji se sastoji od više od sedam tisuća slika. Analizirano je nekoliko dostupnih skupova podataka koji se sastoje od slika i oznaka u obliku graničnih okvira i odabrana su tri najrelevantnija skupa za ovaj problem. Detektor je treniran metodom prijenosnog učenja i metodom bez unaprijed postavljenih težina. Rješenje je potrebno implementirati i na ugradbenu platformu *NVIDIA Jetson* Nano pa je potrebno koristiti i *YOLOv7 tiny* verziju detektora s smanjenim brojem parametara. Nakon treniranja razvijen je algoritam koji kombinira algoritme za detekciju i za praćenje pješaka. Korištenjem *SORT* i *DeepSORT* algoritama za praćenje povećana je brzina izvođenja predloženog algoritma za detekciju i praćenje pješaka čime je omogućeno izvođenje ovog računalno zahtjevnog algoritma u stvarnome vremenu na računalima s ograničenim resursima. Evaluacija rješenja izvedena je na četiri videozapisa koja simuliraju stvarne uvjete.

Ključne riječi: *detekcija pješaka*, *praćenje*, *YOLOv7*, *SORT*, *DeepSORT*

PEDESTRIAN DETECTION AND TRACKING WITHIN AUTONOMUS DRIVING

ABSTRACT

This paper describes the development of a solution for the detection and tracking of pedestrians in the context of autonomous driving. First, a modern way of detecting and tracking pedestrians in digital images is explained. After that, several existing solutions for object detection and tracking are described. The solution is based on the *YOLOv7* detection algorithm and the *SORT* and *DeepSORT* tracking algorithms. The detection algorithm was trained on customized data set consisting of more than seven thousand images. Several available datasets consisting of images and labels in the form of bounding boxes were analyzed and the three most relevant datasets for this problem were selected. The detector was trained using a transfer learning method and a method without pretrained weights. The solution needs to be implemented on the embeded *NVIDIA Jetson Nano* platform, so it is necessary to use the *YOLOv7 tiny* version of the detector with a reduced number of parameters. After training, an algorithm was developed that combines algorithms for detection and tracking of pedestrians. By using *SORT* and *DeepSORT* tracking algorithms, the execution speed of the proposed algorithm for pedestrian detection and tracking has been increased, which enables the execution of this computationally complex algorithm in real time on computers with limited resources. The evaluation of the solution was performed on four videos simulating real world conditions.

Keywords: *pedestrian detection*, *tracking*, *YOLOv7*, *SORT*, *DeepSORT*

ŽIVOTOPIS

Mario Jusup rođen je 1.8.2000. u Našicama. Pohađao je Osnovnu školu „Kralja Tomislava” u Našicama. Nakon završene osnovne škole upisuje Prirodoslovno matematičku gimnaziju u srednjoj školi „Isidora Kršnjavoga” u Našicama. Gimnaziju završava 2019. godine pa iste godine upisuje Preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Završetkom preddiplomskog studija 2022. godine stječe akademski naziv sveučilišni prvostupnik inženjer računarstva i upisuje Sveučilišni diplomski studij računarstvo, DRD modul – informacijske i podatkovne znanosti.

_____

Potpis autora

PRILOZI